UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CAMPUS DE ILHA SOLTEIRA

LIGIA RODRIGUES PRETE

APLICAÇÃO DE REDES DEFINIDAS POR SOFTWARE NO PROCESSO DE GERENCIAMENTO DE ENERGIA NOS SWITCHES DE REDE OPENFLOW

Ilha Solteira
2016

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

LÍGIA RODRIGUES PRETE

APLICAÇÃO DE REDES DEFINIDAS POR SOFTWARE NO PROCESSO DE GERENCIAMENTO DE ENERGIA NOS SWITCHES DE REDE OPENFLOW

Tese apresentada à Faculdade de Engenharia do Campus de Ilha Solteira – UNESP, como parte dos requisitos para obtenção do título de Doutora em Engenharia Elétrica. Área de Conhecimento: Automação.

Prof. Dr. Ailton Akira Shinoda
Orientador

Ilha Solteira
2016

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

P942a

Prete, Ligia Rodrigues.
Aplicação de redes definidas por software no processo de gerenciamento de energia nos switches de rede openflow / Ligia Rodrigues Prete. -- Ilha Solteira: [s.n.], 2016
165 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia. Área de conhecimento: Automação, 2016

Orientador: Ailton Akira Shinoda
Inclui bibliografia

1. Redes definidas por software. 2. Openflow. 3. Floodlight. 4. Geni. 5. Minet.

unesp

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: Aplicação de Redes Definidas por Software no Processo de Gerenciamento de Energia nos Switches de Rede OpenFlow

AUTORA: LÍGIA RODRIGUES PRETE
ORIENTADOR: AILTON AKIRA SHINODA

Aprovada como parte das exigências para obtenção do Título de Doutora em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Prof. Dr. AILTON AKIRA SHINODA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. ANTONIO MARCOS COSSI
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira

Profa. Dra. CHRISTIANE MARIE SCHWEITZER
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira

PROFESSOR DOUTOR EDUARDO COELHO MARQUES DA COSTA
Departamento de Engenharia de Energia e Automação Elétrica / ESCOLA POLITÉCNICA DA USP

Profa. Dra. TEREZA CRISTINA MELO DE BRITO CARVALHO
Departamento de Engenharia de Computação e Sistemas Digitais (PCS) / Universidade Federal de São Paulo - USP

Ilha Solteira, 13 de dezembro de 2016

DEDICATÓRIA

Aos meus queridos pais, Nelson Antonio Prete e Zenaide Rodrigues Prete, e a minha irmã Márcia Donizeth Prete que sempre me apoiaram em todos os estágios de minha vida.

AGRADECIMENTOS

A Deus, que nos concede o direito à vida.

À família, sempre presente nos momentos mais difíceis.

Ao Centro Estadual de Educação Tecnológica Paula Souza que incentivou por meio do Regime de Jornada Integral e do afastamento parcial para a elaboração da pesquisa e do estudo inovador deste trabalho.

Ao Professor Dr. Ailton Akira Shinoda (orientador) e à Professora Dra. Christiane Marie Schweitzer, pelo guiamento na elaboração deste trabalho. Agradeço pela dedicação, ensinamento e por toda paciência no desenvolvimento da pesquisa.

À banca examinadora que analisou o conteúdo deste estudo, contribuindo com sugestões.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP, 2014/06022-9) que apoiou este estudo e a contribuição de GENI - Office Project da National Science Foundation que forneceu infraestrutura de rede para realizar as simulações.

“O insucesso é apenas uma oportunidade para começar de novo com mais inteligência”. Henry Ford

RESUMO

O consumo de energia no setor de Tecnologia da Informação e Comunicação (TIC) tem crescido exponencialmente nos últimos anos, em virtude da quantidade crescente de equipamentos para armazenamento e processamento de dados. O paradigma de Redes Definidas por Software (do inglês, *Software-Defined Networking* - SDN) e a arquitetura OpenFlow estão permitindo uma nova gama de aplicações e serviços para redes. A presente tese apresenta um estudo que aplica tecnologias SDN em um ambiente virtualizado com a federação GENI (*Global Environment for Network Innovation*). Neste trabalho foi desenvolvido um módulo no controlador Floodlight intitulado como Módulo Economia de Energia que emprega um algoritmo denominado MiNet (Mínima Rede) para a construção da Árvore de Extensão Mínima (do inglês, *Minimum Spanning Tree* - MST) sobre os componentes de comutação em redes. Este estudo apresenta três simulações em duas topologias de rede Fat Tree, sendo, uma com dez (FatTree10) e outra com vinte *switches* (FatTree20). Na primeira simulação foi realizada sem o módulo com a configuração padrão do controlador Floodlight para servir de comparação com os resultados de desempenho obtidos nas outras duas simulações. Já a segunda, com o Módulo Economia de Energia incluído no controlador, foi avaliada quanto aos custos iniciais nas ligações entre os *switches*. Na terceira, os custos nas ligações dos *switches* foram alterados para evidenciar que o Módulo Economia de Energia é capaz de recalcular uma nova Árvore de Extensão Mínima sobre os custos fornecidos e assim adaptar-se à rede para uma nova situação de atualização. Por meio de simulações realizadas, considerando somente as ligações entre os *switches*, sendo, quarenta portas Ethernet para a topologia menor e oitenta portas Ethernet para a topologia ampla, de acordo com os resultados alcançados, o módulo incorporado no Floodlight reduziu o consumo de energia final em 35% para a topologia FatTree10 e 32,5% na topologia FatTree20.

Palavras-chave: Redes definidas por software. Openflow. Floodlight. GENI. Minet.

ABSTRACT

Energy consumption in the Information and Communication Technology (ICT) sector has grown exponentially recently, due to the increasing amount of equipment for data storage and processing. The paradigm of Software-Defined Networking (SDN) and OpenFlow architecture are enabling a new range of applications and services for networks. This thesis presents a study that applies SDN technologies in a virtualized environment with the GENI federation (Global Environment for Network Innovation). This paper developed a module in Floodlight controller titled Energy Saving Module employing an algorithm called MiNet (Minimum Network) for the construction of the Minimum Spanning Tree (MST) on the switching components in networks. This study presents three simulations in two network topologies Fat Tree, as it follows, a ten one (FatTree10) and another with twenty switches (FatTree20). In the first simulation, it was performed without the module with the default configuration of Floodlight controller to serve as a comparison with the performance results in the other two simulations. The second, with Module Energy Saver included in the controller, it evaluated the initial costs on the links between switches. In the third, the costs in the connections of the switches were changed to high light that the Energy Savings Module is able to recalculate a new Minimum Spanning Tree on the provided costs and thus adapt the network to a new update situation. Through the performed simulations, considering only the links between switches, as it is, forty Ethernet ports for smaller topology and eighty Ethernet ports for wide topology, according to the achieved results, the embedded module Floodlight reduced the final energy consumption to 35% FatTree10 topology and 32.5% FatTree20 topology.

Keywords: Software-defined networking. Openflow. Floodlight. GENI. Minet.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicativos)
BGP	*Border Gateway Protocol* (Protocolo de Roteamento de Borda)
BonFIRE	*Building Service Testbeds on Future Internet Research and Experimentation* (Construindo Testes de Serviço na Pesquisa Experimental para Internet do Futuro)
CARPO	*Correlation-Aware Power Optimization* (Otimização de Energia de Correlação)
CO_2	Dióxido de carbono
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
CPU	*Central Processing Unit* (Unidade Central de Processamento)
CREW	*Cognitive Radio Experimentation World* (Experimento Mundial de Rádio Cognitivo)
DERI	*Digital Enterprise Research Institute* (Instituto de Pesquisa de Empresa Digital)
DHCP	*Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de *Host*)
D-ITG	*Distributed Internet Traffic Generator* (Gerador de Tráfego para Internet Distribuído)
DNS	*Domain Name System* (Sistema de Nomes de Domínios)
ECODANE	*Reducing **E**nergy **CO**nsumption in **DA**ta Center **Ne**tworks* (Redução do Consumo de Energia em Redes Data Center)
EUA	*United States of America* (Estados Unidos da América)
FIBRE	*Future Internet Testbeds Experimentation between Brazil and Europe* (Ambiente de Experimentação para Internet do Futuro entre o Brasil e Europa)
FIRE	*Future Internet Research and Experimentatio*n (Pesquisa Experimental para Internet do Futuro)
GENI	*Global Environment for Network Innovation* (Ambiente Global para a Inovação de Rede)
HP	Hewlett-Packard
ICMP	*Internet Control Message Protocol* (Protocolo de Mensagens de Controle da Internet)

ICT	*Information and Communication Technology* (Tecnologia da Informação e Comunicação)
IDC	*International Data Corporation* (Corporação de Dados Internacional)
INCT	Institutos Nacionais de Ciência e Tecnologia
IF	Internet do Futuro
IP	*Internet Protocol* (Protocolo de Internet)
IPv4	*Internet Protocol version* 4 (Protocolo de Internet versão 4)
L3S	L3S *Research Center* (Centro de Pesquisa L3S)
LERO	*The Irish Software Engineering Research Centre* (Centro de Pesquisa de Engenharia de Software de Irish)
LIP6	*Laboratoire d'Informatique de Paris VI* (Laboratório de Informática de Paris VI)
LLDP	*Link Layer Discovery Protocol* (Protocolo de Descoberta da Camada de Enlace)
LNCC	Laboratório Nacional de Computação Científica
MAC	*Media Access Control* (Controle de Acesso ao Meio)
Mbps	Megabit por segundo
MiNet	*Minimum Network* (Mínima Rede)
MST	*Minimum Spanning Tree* (Árvore de Extensão Mínima)
NAT	*Network Address Translation* (Tradução de Endereço da Rede)
NEC	Nippon Electric Company
NSF	*National Science Foundation* (Fundação de Ciência Nacional)
OFELIA	*OpenFlow in Europe: Linking Infrastructures and Applications* (OpenFlow na Europa: Ligando Infraestruturas e Aplicações).
OMF	*cOntrol and Management Framework* (Framework de Gerência e Controle)
ONF	*Open Networking Foundation* (Fundação de Rede Aberta)
ORBIT	*Open-Access Research Testbed for Next-Generation Wireless Networks* (Ambiente de Pesquisa de Acesso Aberto para Próxima Geração de Redes Sem Fio)
OSPF	*Open Shortest Path First* (Abre Menor Rota Primeiro)
PC	*Personal Computer* (Computador Pessoal)
PCAP	*Packet Capture* (Captura de Pacote)
pcvm	*personal computer virtual machine* (máquina virtual no computador pessoal)
PLC	*Power Line Comunication* (Comunicação de Linha de Força)

PLC	*PlanetLab Central Software* (Servidor Centralizado PlanetLab)
PoPs	Pontos de Presença
pps	*Packets per second* (pacotes por segundo)
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro
QoS	*Quality of Service* (Qualidade de Serviço)
REST	*Representational State Transfer* (Transferência de Estado Representacional)
RNP	Rede Nacional de Ensino e Pesquisa
RSpec	*Resource Specification* (Especificação de Recursos)
s	segundo
SDH	*Synchronous Digital Hierarchy* (Hierarquias de Multiplexação Digital)
SDN	*Software-Defined Networking* (Redes Definidas por Software)
SFA	*Slice-Based Facility Architecture* (Arquitetura Federativa Baseada em Fatias)
SNAC	*Simple Network Access Control* (Controle de Acesso de Rede Simples)
SNMP	*Simple Network Management* Protocol (Protocolo Simples de Gerência de Rede)
SSL	*Secure Socket Layer* (Camada de Suporte à Segurança)
TIC	Tecnologia da Informação e Comunicação
TCP	*Transmission Control Protocol* (Protocolo de Controle e Transmissão)
TCP/IP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão) / *Internet Protocol* (Protocolo de Internet)
U.Waterloo	University of Waterloo
UENF	Universidade Estadual do Norte Fluminense
UERJ	Universidade do Estado do Rio de Janeiro
UFC	Universidade Federal do Ceará
UFES	Universidade Federal do Espírito Santo
UFF	Universidade Federal Fluminense
UFG	Universidade Federal de Goiás
UFPA	Universidade Federal do Pará
UFPE	Universidade Federal de Pernambuco
UFRGS	Universidade Federal do Rio Grande do Sul
UFRJ	Universidade Federal do Rio de Janeiro
UFRN	Universidade Federal do Rio Grande do Norte
UFSCar	Universidade Federal de São Carlos
UNICAMP	Universidade Estadual de Campinas

UNIFACS	Universidade Salvador - Laureate International Universities
UNIRIO	Universidade Federal do Estado do Rio de Janeiro
USP	Universidade de São Paulo
UtahDDC	Utah Downtown Data Center
VLAN	*Virtual Local Area Network* (Redes Locais Virtuais)
VNF	*Virtual Network Filter* (Filtro de Rede Virtual)
XML	*eXtensible Markup Language* (Linguagem Extensível de Marcação)

SUMÁRIO

1 INTRODUÇÃO

De modo em geral, a necessidade de se adotar medidas ecologicamente corretas, torna-se imprescindível, para ampliar o equilíbrio ao ecossistema do planeta (SILVA et al., 2010).

A Tecnologia da Informação e Comunicação é conhecida como o conjunto de todas as atividades oriundas dos recursos de computação. Ela é utilizada por empresas que desejam melhorar seus processos com a intenção de modernizar seus negócios e, consequentemente, torná-las mais competitivas. No entanto, a implantação generalizada de TIC tem gerado efeitos indesejados devido, principalmente, ao consumo desnecessário de recursos naturais do planeta (DROUANT et al., 2014).

Para controlar o consumo excessivo de recursos, pode-se recorrer a uma prática sustentável conhecida como TIC Verde (Tecnologia da Informação e Comunicação Verde), que tem como objetivo utilizar a tecnologia da informação de forma consciente e inteligente com o intuito de diminuir o consumo de energia através de uma melhor utilização dos recursos. Tais práticas promovem melhorias nas atividades das empresas, garantem uma redução de custo e colaboram com a preservação do meio ambiente.

De acordo com Alves (2010), o setor de TIC é responsável por 1,4% do total da emissão global de CO_2 (dióxido de carbono). Destes, os PCs (*Personal Computer*) e monitores contribuem com 57%, os servidores e infraestrutura de TIC com 34% e as impressoras com 9%. Segundo a *Internacional Energy Agency* (2013), cerca de 81,63% de toda a matriz energética global advém de energia fóssil (carvão, petróleo e gás natural). Para gerar eletricidade, a energia fóssil primária precisa ser queimada e assim libera grandes quantidades de dióxido de carbono na atmosfera, contribuindo para o aumento do efeito estufa e aquecimento global. O IDC (*International Data Corporation*), empresa especializada em inteligência de mercado, consultoria e conferências no ramo de Tecnologia da Informação e Telecomunicações, realizou um estudo e destacou que o Brasil tem capacidade de reduzir aproximadamente 27% da emissão de CO_2 até 2020. Tal redução é baseada no índice de sustentabilidade de TIC, que é uma forma criada pelo IDC para classificar as nações do G-20[1], tomando como base a sua capacidade de reduzir emissões de gases estufa através do uso de TIC.

A emissão de CO_2 na área de TIC acontece principalmente devido ao mau gerenciamento de energia dos equipamentos encontrados nos *switches*[2] dos países

[1]Grupo dos 20 (ou G-20) é um grupo formado pelos ministros de finanças e chefes dos bancos centrais das 19 maiores economias do mundo mais a União Europeia.
[2]*Switches* são comutadores utilizados em redes de computadores para reencaminhar pacotes entre os diversos nós.

industrializados. A sociedade e as empresas tentam encontrar alternativas para melhorar a eficiência energética da área de TIC e diminuir o impacto no meio ambiente (SIVARAMAN et al., 2014).

O paradigma de Redes Definidas por Software e a infraestrutura OpenFlow oferecem um caminho para vencer esse desafio, por meio de uma solução que seja implantada de forma gradativa em redes de produção (MCKEOWN, 2008). Dessa forma, além de atender os problemas de gerenciamento de energia em redes de computadores, o paradigma SDN nos permite atender as necessidades das novas aplicações de rede, resolvendo problemas associados a requisitos de escalabilidade, mobilidade, segurança, entre outros, por meio de uma arquitetura de rede programável.

Redes Definidas por Softwares são conceituadas como um paradigma emergente para comunicação de dados que está mudando completamente a configuração do plano de controle e transmissão de dados nas redes. A arquitetura SDN permite a criação de novos serviços e protocolos abertos que beneficiam os administradores, por meio da otimização nas redes, permitindo que aplicações se adaptem dinamicamente em suas infraestruturas, para suprir as próprias necessidades. Uma utilização interessante do paradigma SDN é a redução do consumo de energia em redes de larga escala (XIE et al., 2015; WANG et al., 2016).

O protocolo OpenFlow (MCKEOWN et al., 2008; SHARMA et al., 2013) é um padrão aberto, sendo a tecnologia mais difundida da arquitetura SDN. Um elemento de rede habilitado para OpenFlow delega a decisão sobre como lidar com os pacotes de entrada para um elemento externo, denominado controlador. Os dispositivos de encaminhamento e o controlador remoto comunicam-se por meio de um canal seguro. Um *switch* OpenFlow é equipado com uma tabela de base de informação de encaminhamento, armazenando regras de correspondência para os pacotes de entrada (por exemplo, encaminhar para uma porta, descartar o pacote, ou modificar seu cabeçalho) e contadores. Se um pacote de entrada corresponde a uma regra de encaminhamento, a ação correspondente é tomada e os contadores são atualizados. Quando um pacote não corresponde a nenhuma regra, ele é enviado para o controlador, onde a lógica da aplicação formula uma nova regra que será implantada no dispositivo para encaminhar o tráfego de entrada que corresponde à mesma regra (OPENFLOW, 2011). OpenFlow está sendo adotado por um número crescente de administradores de rede de grande porte.

1.1 OBJETIVO

O objetivo deste trabalho é analisar e propor uma solução para a problemática do consumo de energia e do impacto gerado pelas tecnologias da informação, em particular, na topologia Fat Tree utilizada em *data centers*[3]. Este estudo descreve o projeto e a implementação de um módulo no controlador SDN Floodlight que institui um algoritmo inovador e de código fonte aberto para que outros pesquisadores possam aprimorar a construção da Árvore de Extensão Mínima entre os comutadores de rede, com o objetivo de aumentar o seu aspecto ecológico. O ambiente laboratorial será a federação GENI que fornecerá infraestrutura para análise em redes OpenFlow.

1.2 EFICIÊNCIA ENERGÉTICA - MOTIVAÇÃO

O *data center* tem como objetivo proporcionar uma infraestrutura computacional confiável e escalável para serviços massivos da Internet. Para atingir essas propriedades, que consomem energia resultando em custos operacionais, tem-se estimulado o interesse em melhorar a sua eficiência. A maioria dos esforços são focados em servidores e refrigeração, que representam cerca de 70% do orçamento total de energia em um *data center*. As melhorias incluem componentes (baixo consumo de energia de CPUs - *Central Processing Unit*, fontes de alimentação mais eficientes, entre outros) e *software* (*kernel*, virtualização e *Smart Cooling*) (GRUNWALD et al., 2000; PATEL et al., 2013). Analisando os itens do *data center*, a rede consome de 10 a 20% do total de energia (GREENBERG et al., 2009).

A economia de energia ao desligar um *switch* individual pode parecer insignificante, no entanto, em um amplo *data center*, que hospeda centenas de milhares de servidores e é composto por dezenas de milhares de *switches* implantados, consequentemente haverá um ganho expressivo, uma eficiência significativa. A economia de energia depende dos padrões de tráfego, do nível desejado de redundância do sistema e do tamanho do próprio *data center*. Os experimentos realizados por Heller et al. (2010) e os relatos em outras literaturas mostram que, em média, uma economia de 25 a 40% de energia na rede em *data centers* é viável. Além disso, a redução do consumo de energia dos dispositivos de rede resulta em uma redução proporcional nos custos de resfriamento.

[3]*Data centers* são centro de dados onde estão concentrados os equipamentos de processamento e armazenamento de dados de uma empresa ou organização.

1.3 CONTRIBUIÇÕES DO TRABALHO

O módulo desenvolvido intitulado como “Módulo Economia de Energia” no controlador OpenFlow (Floodlight) terá um benefício considerável se comparado com as soluções mais especializadas e estará disponível para um público maior de administradores de rede. A abordagem visa reduzir o impacto ambiental e as despesas das redes, desativando as conexões que apresentam ciclos entre os comutadores OpenFlow. Adota-se uma estratégia com a adaptação e melhoramento de algoritmos que estará disponível com código fonte aberto, na qual administradores de redes possam adaptar melhores configurações na rede para aumentar a facilidade de escolha, evitando os ciclos e permitindo a investigação sobre os métodos rápidos de transferências em caso de falhas. Os algoritmos existentes possuem código fonte fechado pelos fabricantes de *switches* tradicionais e, sendo assim, não podem ser alterados para as novas tendências em Internet do Futuro.

A Internet do Futuro irá possuir mecanismos de adaptação do uso de energia para oferecer uma comunicação eficiente, na transmissão de dados e no gerenciamento de rede, a baixos níveis energéticos.

Os resultados deste estudo poderão ser utilizados para inovação e desenvolvimento da ciência computacional e da comunicação entre dispositivos eletrônicos. O meio industrial e comercial da área de redes de comunicação, do mesmo modo, poderá ser beneficiado com este trabalho, visto que novos equipamentos ou *softwares* poderão surgir e, assim, serem utilizados mundialmente.

Com a divulgação do estudo por meio de participações em congressos e publicações em periódicos, esta instituição ao mesmo tempo, se fortalecerá como difusora de conhecimento, avanço tecnológico e pesquisa científica.

1.4 TRABALHOS RELACIONADOS

Otimização do consumo de energia elétrica em redes é um tema que está recebendo ampla relevância. A necessidade de minimizar os custos operacionais de infraestruturas de TIC é considerada importante, tal como a conectividade de rede, que, até o momento, era considerada intocável, devido ao seu papel essencial.

Nesta subseção, serão apresentadas algumas soluções de gerenciamento de energia que compartilham um objetivo incomum, ou seja, reduzir o consumo de energia em *datacenters* por

meio da utilização de algoritmos. Existem vários projetos desenvolvidos e em desenvolvimento, a fim de melhorar a eficiência energética (JARSCHEL; PRIES, 2012).

Heller et al. (2010) apresenta ElasticTree, um gerenciador de energia para ampla rede, que ajusta dinamicamente o conjunto de elementos ativos (*enlaces* e *switches*), para satisfazer as mudanças de cargas de tráfego do *data center*. Neste projeto, compararam-se várias estratégias para encontrar subconjuntos da rede com energia mínima por meio de uma série de padrões de tráfego. O ElasticTree foi implementado e analisado seguido por um protótipo de teste construído com a produção de *switches* OpenFlow, a partir de três fornecedores de rede. Além disso, foram analisadas as vantagens e desvantagens entre eficiência energética, desempenho e robustez, com registros reais de um site *e-commerce* em produção. Os resultados demonstraram que, para altas cargas de trabalho do *datacenter*, ElasticTree pode-se economizar até 50% de energia da rede, ao mesmo tempo que mantém a capacidade de lidar com picos de tráfego.

Wang et al. (2012) propõe CARPO (*Correlation-Aware Power Optimization*), um algoritmo de otimização de energia de correlação que consolida dinamicamente o tráfego de dados para um pequeno conjunto de *links* e *switches* em um *data center* e logo desliga dispositivos de rede não utilizados para a economia de energia. CARPO foi projetado com base na análise do rastreio de um *data center* realístico que demanda largura de banda de diferentes picos de fluxos de dados não exatamente simultâneos. Além disso, CARPO integra a consolidação do tráfego com a adaptação da taxa de *link* para a economia de energia. Foi implementado em um ambiente de *hardware* composto por dez *switches* virtuais OpenFlow (configurado com quarenta e oito portas) e oito servidores. CARPO pode economizar até 46% de energia de rede em um *data center*.

Huong et al. (2011) e Thanh et al. (2012) dispõe um framework denominado ECODANE (*Reducing **E**nergy **CO**nsumption in **DA**ta Center **NE**tworks*) que reduz o consumo de energia no *data center* baseado em engenharia de tráfego. Ele se concentra em otimizar o consumo de energia de componentes de rede por meio da concepção de um sistema de controle de rede inteligente que adapta dinamicamente ao conjunto de componentes de rede ativos que correspondem ao tráfego total que passa pelo *data center*. O módulo otimizador está acompanhado por um módulo roteador de balanceamento de carga para garantir a disponibilidade do *data center*. O sistema proposto foi construído usando o emulador virtual de rede, Mininet, *switches* OpenFlow e o controlador NOX, e avaliado no ambiente de *hardware* usando *switches* OpenFlow baseados em NetFPGA. Os resultados experimentais demonstraram

que pode ser alcançada uma economia de energia entre 10% a 35%, ajustando dinamicamente o número de *switches* ativos, portas e *linecards*.

Riekstin et al. (2016) mostra um método capaz de orquestrar diferentes funcionalidades de eficiência energética, considerando as possíveis combinações e conflitos entre elas, bem como a melhor opção para uma dada carga de trabalho e características da rede. No método proposto, as políticas de negócios são refinadas até o nível de rede de modo a trazer as diretivas de negócios para dentro da operação da rede, e uma Função de Utilidade é usada para combinar requisitos de eficiência energética e desempenho. Uma Árvore de Decisão capaz de determinar o que fazer em cada cenário é implementada em um ambiente de Redes Definidas por *Software*. O método proposto foi validado com diferentes experimentos, testando-se a Função de Utilidade, checando a economia adicional de energia ao combinar funcionalidades, a interpolação da Árvore de Decisão e aspectos de dinamicidade. A orquestração mostrou-se válida para resolver o problema de como encontrar a melhor combinação de funcionalidades para um determinado cenário, obtendo economias adicionais de energia devido à combinação de funcionalidades, além de garantir uma operação sem conflitos.

A literatura relata aplicações e algoritmos com o intuito de reduzir o consumo de energia em *data centers*. Este trabalho visa contribuir com a construção de algoritmos com código fonte aberto, tampouco desenvolvidos, com a finalidade de criar sobreposições livres de ciclo em redes, desativando conexões que apresentam laços entre os *switches* OpenFlow para reduzir o impacto ambiental em *data centers*.

1.5 ORGANIZAÇÃO DO TEXTO

Este estudo apresenta informações sobre a problemática do consumo de energia, o impacto gerado no setor de TIC das empresas e a prática sustentável conhecida como TIC Verde (seção 1). Relata o objetivo proposto no trabalho (subseção 1.1). Apresenta os parâmetros satisfatórios para eficiência energética (subseção 1.2). Descreve a contribuição do trabalho para redes OpenFlow (subseção 1.3). Mostra os trabalhos relacionados na literatura (subseção 1.4). Assim, o trabalho está organizado em seções, incluindo esta introdução, conforme os tópicos a seguir:

a) **Seção 2:** mostra uma breve descrição da Internet do Futuro, mostrando os desafios relacionados e a pesquisa experimental no Brasil e no Mundo, além de citar suas arquiteturas;
b) **Seção 3:** apresenta uma introdução sobre virtualização com enfoque em redes;

c) **Seção 4:** define o padrão OpenFlow, bem como, seus componentes, protocolo e controlador, além de mostrar o seu funcionamento, aplicações e as novas tendências;
d) **Seção 5:** aborda as Redes Definidas por Software, assim como, sua arquitetura, os elementos programáveis das redes SDN, controladores e suas aplicações;
e) **Seção 6:** será exposto o Módulo Economia de Energia no controlador Floodlight, suas características e o fluxo de execução para o gerenciamento de energia nos *switches* de redes OpenFlow;
f) **Seção 7:** é exibido no ambiente de experimentação GENI a plataforma de testes, resultados e discussão de três simulações realizadas em duas topologias, sendo, uma com dez e outra com vinte *switches* para medir as ações de eficiência energética do Módulo Economia de Energia;
g) **Seção 8:** finalmente mostra as discussões finais do estudo;
h) **Seção 9:** apresenta as conclusões, sugestões para trabalhos futuros e lista de publicações realizadas.

2 INTERNET DO FUTURO

Atualmente, na comunidade acadêmica, existe um amplo consenso de que a Internet atual possui várias limitações relacionadas com escalabilidade, suporte a redes móveis de vários tipos (ad-hoc, multi-hop e mesh), mobilidade, consumo de energia, segurança, entre outros (JAIN, 2006). A partir disso, nesta seção, serão comentados os principais desafios que estão relacionados à Internet do Futuro, bem como, os projetos a respeito de sua investigação, como o caso do GENI (*Global Environment for Network Innovation*), financiado pela NSF (*National Science Foundation*), FIRE (*Future Internet Research and Experimentation*), pela União Europeia e por iniciativas brasileiras.

2.1 DESAFIOS RELACIONADOS À INTERNET DO FUTURO

A definição básica da arquitetura da Internet foi desenvolvida há aproximadamente trinta anos. Neste período, tem-se aprendido sobre redes de computadores e encaminhamento de pacotes. Além disso, durante este mesmo período, a Internet vem lidando com contínuas adaptações, causadas pelo seu rápido crescimento e pela quantidade de usuários e aplicações habilitadas sobre sua arquitetura. Essas adequações demonstram que o projeto inicial já não se ajusta às necessidades atuais na rede. Ainda, a arquitetura atual da Internet já apresenta inúmeros problemas ainda não solucionados, impedindo o atendimento dos requisitos destas novas aplicações e serviços (FARIAS et al., 2011).

Diversos especialistas em redes de computadores chegaram a um consenso de que agora consideram extremamente importante realizar estudos de arquiteturas alternativas para Internet do Futuro como uma maneira realmente eficiente de resolver muitos dos problemas prementes que atualmente afligem a Internet. A seguir, serão apresentados alguns dos principais desafios, que, segundo Paul et al. (2011), a nova arquitetura da Internet deve atender.

2.1.1 Suporte às inovações de tecnologias de rede

A Internet vigente é projetada para tirar proveito de uma ampla gama de tecnologias de comunicação em rede. No entanto, hoje há vários desafios, entre elas, as tecnologias sem fio e as novas soluções ópticas.

Atualmente, a mobilidade está na “borda” da rede, mas, os mecanismos para torná-la mais eficiente não funcionam de maneira satisfatória, principalmente nas questões relacionadas

à eficiência energética, mudança de ponto de acesso e aplicações. Nesse cenário, identificam-se os seguintes desafios da Internet do Futuro para suporte às tecnologias sem fio:

a) Suportar a mobilidade como o objetivo primordial em sua construção. Os nós devem ser capazes de modificar seu ponto de acesso na Internet e, mesmo assim, continuarem sendo alcançáveis;
b) Oferecer meios adequados para uma aplicação de descobrir as características dos mais variados tipos de enlace de rede sem fio e se adaptar a eles, de maneira a prover as eficiências energéticas destes nós;
c) Facilitar o processo pelo qual os nós móveis, fisicamente próximos, descubram-se uns aos outros.

A rede de transporte óptica é outra tecnologia revolucionária. Logo identificam os desafios para Internet do Futuro explorar a emergente capacidade óptica:

a) Autorizar os usuários a empregarem essas novas capacidades de transporte óptico, incluindo uma melhor confiabilidade por meio de diagnósticos entre camadas;
b) Permitir que os nós que são dinamicamente configuráveis habilitem a camada eletrônica para o acesso dinâmico de usuários;
c) Abranger *softwares* de controle e gerenciamento que permitam à rede formada por nós dinamicamente reconfiguráveis ser configurada por aplicações que necessitem de uma ampla quantidade de recursos de rede, tal como largura de banda.

2.1.2 Segurança e robustez

O ensejo que mais instigou a comunidade acadêmica a pensar em uma reestruturação da Internet foi a possibilidade de ter uma rede com amplos avanços na segurança e na robustez. Na Internet atual, não há nenhuma abordagem, realmente abrangente para tratar questões de segurança. Ela possui vários mecanismos, mas não uma “arquitetura segura” e muito menos um conjunto de regras determinando como esses mecanismos devem ser compatíveis para se obter uma apropriada segurança de maneira geral. A segurança em redes e, principalmente a da Internet, assemelha-se a um conjunto crescente de correções, extremamente suscetível a falhas.

Para elaboração de uma rede mais robusta e segura, identificam-se os seguintes desafios:

a) Qualquer conjunto de nós deve ser capaz de se comunicar entre si com alta confiabilidade e nós maliciosos ou defeituosos não podem ser capazes de interromper esta comunicação;

b) A segurança e robustez devem ser estendidas por meio de suas camadas, uma vez que a segurança e confiabilidade de um usuário final dependem da robustez, tanto nas camadas de comunicação quanto nas aplicações distribuídas.

2.1.3 Eficiência energética na comunicação

Há, hoje, uma preocupação mundial pelo desenvolvimento de tecnologias que amenizem o consumo de energia, especialmente em redes, como as de sensores sem fio e as de dispositivos móveis Wi-Fi, cujos recursos energéticos são fatores relevantes em sua comunicação (JOSEPH; LEWIS, 2008). No entanto, a Internet atual não possui mecanismos que levam em consideração a comunicação com requisitos de eficiência de energia, de modo a adaptar o seu consumo mediante a observação do meio. Ou ainda, que adapte o uso de energia para oferecer uma comunicação eficiente, na transmissão de dados e no gerenciamento de rede, a baixos níveis energéticos.

Desse modo, frente aos desafios para o projeto da Internet do Futuro, deve-se:

a) Fornecer meios para uma eficiência energética generalizada tanto para dispositivos sem fio quanto os com fio;
b) Ampliar tecnologias com o foco no uso eficiente da energia elétrica;
c) Abranger, em sua arquitetura, mecanismos que ofereçam uma comunicação eficiente, tanto na transmissão de dados como no gerenciamento de rede;
d) Prover uma arquitetura para Internet do Futuro energeticamente otimizada.

2.1.4 Separação de identidade e endereço

Na Internet, um sistema é identificado pelo seu endereço IP (*Internet Protocol*). Como decorrência, quando o sistema mudar a localização do ponto da sua interligação, seu endereçamento ainda pode ser alterado, ou seja, o endereço IP, neste caso, ao mesmo tempo, tem o papel de identificador e localizador. Devido a isso, os nós móveis se tornaram um desafio na Internet atual. Por essas particularidades, torna-se difícil iniciar a comunicação com um sistema móvel.

Esse é um problema bem conhecido na Internet e há um número considerável de experimentos e propostas para resolver esse problema, incluindo: IP móvel, infraestrutura da Internet (STOICA et al., 2004), protocolo de identificação do nó (MOSKOWITZ et al., 2005), entre outros (BALAKRISHNAN et al., 2004).

De tal modo, para a Internet do Futuro existem os seguintes desafios, a saber:

a) Aplicar soluções que permitam a localização global de um determinado nó, por meio da utilização de um sistema de endereçamento global;
b) Definir novas formas de localização e identificação, além do desenvolvimento de endereçamentos coesos às necessidades da rede.

2.1.5 Qualidade de serviço e qualidade de experiência

O IP, não orientado à conexão, dificulta qualquer aplicação de garantia de QoS (*Quality of Service*). Além disso, a característica de encaminhamento de pacote baseado em melhor esforço faz com que qualquer abordagem relacionada a questões de reserva de recursos ou prioridade interfira no funcionamento estabelecido para Internet atual.

Portanto, embora a qualidade de serviço tenha sido extremamente pesquisada na comunidade acadêmica ainda não há um modelo claro de como distintos níveis de qualidade devem ser aplicados e de que forma ele se integrar arquitetura de rede atual.

Dessa forma, são desafios para arquitetura da Internet do Futuro:

a) Permitir uma variedade de garantias levando em consideração tanto a qualidade da aplicação na rede como a qualidade de experiência do usuário;
b) Novos métodos de encaminhamentos de pacote baseados nas características das aplicações, principalmente as de tempo real.

2.1.6 Separação entre os planos de controle, gerenciamento e dados

Na Internet atual, os planos de controle, gerenciamento e dados são unificados, ou seja, mensagens de controle, por exemplo, mensagens de conexão TCP (*Transmission Control Protocol*) ou mensagens de gerenciamento SNMP (*Simple Network Management Protocol*) seguem no mesmo canal em que trafegam os dados. Como consequência, ocorre a possibilidade de um significante risco de segurança dos dados de controle, além do desperdício de recursos da rede.

Uma vantagem desta separação de planos é a adoção de novas tecnologias de plano de dados pela arquitetura de rede como: um comprimento de onda; quadro SDH (*Synchronous Digital Hierarchy*); ou uma linha de transmissão de energia – PLC (*Power Line Comunication*). Logo, a separação entre os planos deve ser parte integrante desta próxima geração da arquitetura da Internet.

2.1.7 Isolamento

Para algumas aplicações críticas, o usuário demanda isolamento em um ambiente compartilhado como na Internet atual. Isolamento significa garantir que o desempenho desta aplicação crítica não seja afetado por outras aplicações que queiram compartilhar o mesmo recurso.

Com o uso da virtualização, o isolamento tornou-se um fator ainda mais importante, principalmente para virtualização de redes, onde um roteador ou uma infraestrutura de rede é totalmente virtualizada, de forma que o encaminhamento de pacotes não pode, de maneira alguma, sofrer interferências, ou causá-las em outras infraestruturas.

Assim, a próxima geração da Internet deve prover em sua arquitetura um modo eficiente de prover uma combinação programável de isolamento e compartilhamento às suas aplicações e serviços.

2.1.8 Mobilidade

Com a crescente e diversificada quantidade de serviços na Internet, impulsionada principalmente pelo aumento do acesso de dispositivos sem fio, amplia-se a necessidade de mobilidade pelos usuários.

Com isso, a forma de comunicação estabelecida originalmente pela arquitetura da Internet atual como conexão ponto-a-ponto e entrega imediata, faz com que esses tipos de usuários não sejam atendidos satisfatoriamente, principalmente, por causa de uma questão comum relacionada à mobilidade, que são os *handovers*[4], criados com o objetivo de evitar que os nós móveis tenham a perda das suas conexões ativas. Os protocolos da Internet atual não preveem o tratamento desse comportamento.

Com isso, a Internet do Futuro enfrenta o grande desafio de como permitir a movimentação do nó entre diferentes pontos de acesso sem que as conexões ativas sejam perdidas.

[4]*Handover* ou *Handoff* é o procedimento empregado em redes sem fios para tratar a transição de uma unidade móvel de uma célula para outra de forma transparente ao utilizador.

2.1.9 Escalabilidade

Com o aumento exponencial do número de estações ou sistemas conectados à Internet, alguns componentes da arquitetura atual têm sofrido com problemas de escalabilidade.

Esse é o caso do sistema de roteamento, que sofre com o aumento e as atualizações frequentes de suas tabelas. Ainda há a questão dos endereçamentos que não são capazes de suportar todos os elementos conectados à rede, como é o caso do IPv4 (*Internet Protocol version* 4).

Portanto, a Internet do Futuro terá o desafio de manter o sistema global de roteamento escalável, além de novos protocolos que mantenham essa escalabilidade, mesmo com o aumento do espaço de endereçamento, além de novas formas de endereçamento para evitar a escassez do recurso.

2.2 PESQUISA EXPERIMENTAL EM INTERNET DO FUTURO NO BRASIL E NO MUNDO

Na comunidade de pesquisa em redes, ao longo do tempo atentou-se ao crescimento de problemas da Internet, aumentando o interesse em estudar os desafios da Internet do Futuro e, com isso, modelar a arquitetura que conduzirá a uma nova geração da Internet.

No entanto, essas inovadoras propostas e considerações teóricas na direção dessas soluções deveriam ser suportadas por uma infraestrutura real de rede experimental e testadas em ambientes de larga escala. Essas instalações experimentais desempenhariam o papel de rede para experimentação ou ambiente de teste[5], possibilitando experimentos para a prova de conceitos de novas arquiteturas, protocolos, tecnologias e serviços.

Uma coexistência com a rede de tráfego de produção é essencial para observação e captura de certos aspectos e fenômenos perceptíveis apenas em instalações operacionais e assim permitir que sejam avaliados os seus impactos sobre a sociedade e a economia.

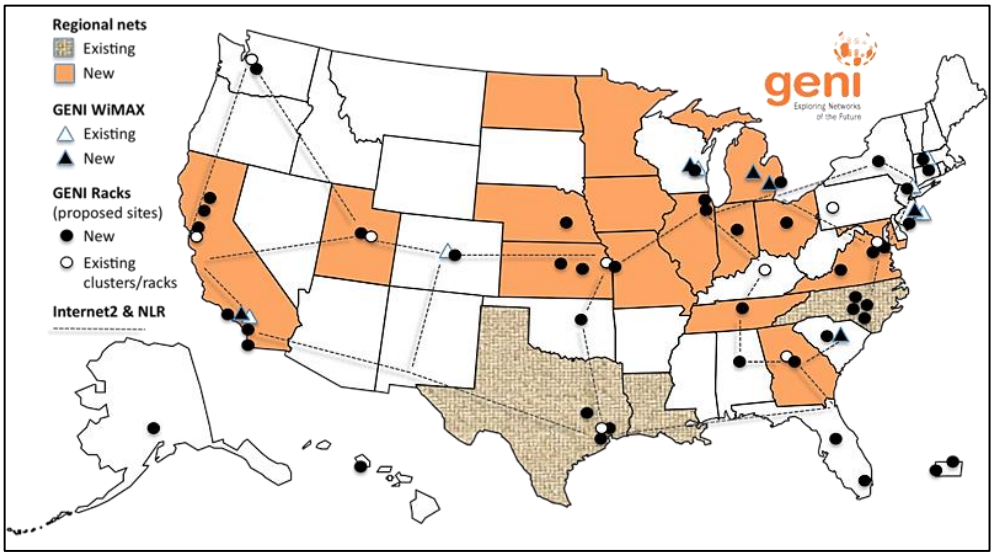
Ressalvando essas necessidades, algumas iniciativas em pesquisa na Internet do Futuro começaram a surgir, em diferentes partes do mundo, conforme detalhamento a seguir.

[5] Encontram-se nas literaturas pelo termo de *testbed*.

2.2.1 GENI

GENI, patrocinado pela NSF desde 2005, é a principal iniciativa norte-americana em investigação e experimentação em Internet do Futuro, conforme exibido na Figura 1. O GENI é um conjunto de infraestruturas de redes para experimentação, dos mais variados modelos tais como: sem fio, óptico e elétrico.

Figura 1 – Federação GENI



Fonte: GENI (2016).

O GENI é um portal de infraestrutura (conhecido como *testbed*) que disponibiliza um ambiente laboratorial para redes e sistemas distribuídos para ensino e pesquisa com múltiplos *testbeds*. O laboratório virtual possibilita pesquisas sobre o futuro das redes de grande porte, criando oportunidades de compreensão, inovação e transformação das redes globais e suas interações com a sociedade. O GENI provê:

a) Suporte à experimentação em larga escala em uma infraestrutura compartilhada, heterogênea e altamente equipada;
b) Alta programabilidade da rede, promovendo inovação nas áreas de redes, segurança, tecnologia, serviços e aplicações;
c) Ambientes colaborativos para academia, indústria, governo e público em geral, catalisando descobertas e inovação.

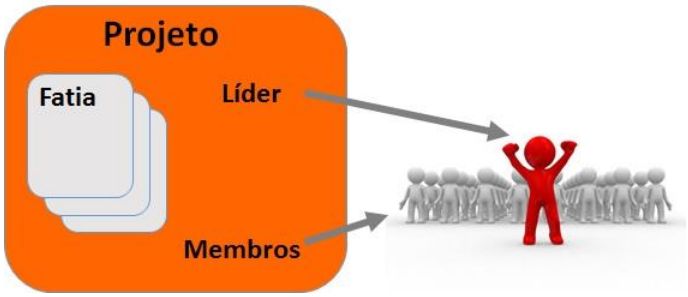
2.2.1.1 Funcionamento

Para iniciar os estudos no ambiente GENI faz-se necessário ser filiado a uma instituição parceira do projeto ou, caso não exista, preencher um formulário especificando o projeto e a instituição de vínculo.

Logo após a criação da conta, os tópicos a seguir precisam ser definidos para começar os testes de experimentos:

a) **Projeto:** organiza pesquisas em GENI, contendo as pessoas e suas experiências. Ele é criado e liderado por um único indivíduo responsável: o líder. Um projeto pode ter muitos membros e este pode ser membro de vários projetos. O líder é em última instância o responsável por todas as ações dos membros do projeto em seu contexto. Membros poderão ter privilégios de líder para a criação de projetos. Apenas professores e membros superiores de uma empresa podem ser líderes. A Figura 2 mostra o esquema de um projeto;

Figura 2 – Elementos de um projeto no GENI



Fonte: Adaptado de GENI (2016).

b) **Fatia:** GENI é um *testbed* compartilhado, ou seja, múltiplos membros podem estar executando várias experiências ao mesmo tempo. Isto é possível por causa do conceito de fatia. Uma fatia GENI é:
   - A unidade de isolamento para os experimentos. Um experimento GENI situa-se em uma fatia. Somente experimentadores que são membros de uma fatia podem fazer alterações nos experimentos,
   - Um contêiner para os recursos usados em um experimento. Membros adicionam recursos (computação, *links* de rede, etc.) para fatias e executam experimentos que usam esses recursos. Um experimento somente usa recursos em sua fatia,

- Uma unidade de controle de acesso. O experimentador que cria uma fatia pode determinar quais serão os membros que terão acesso a fatia do projeto. O líder do projeto é automaticamente um membro de todas as fatias nele criado.

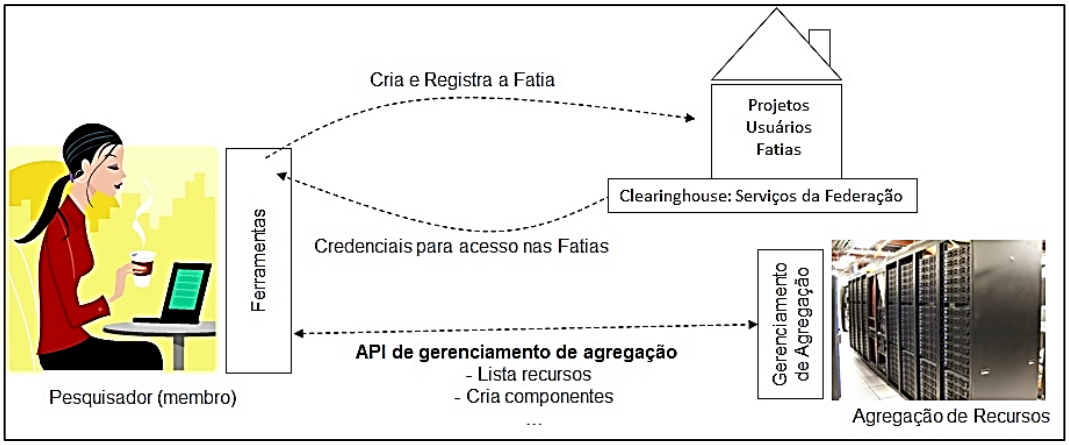
c) **Agregação**: um agregado fornece recursos para experimentos GENI. Por exemplo, um *rack* GENI em uma universidade é um agregado; membros podem solicitar recursos deste agregado e adicioná-los à sua fatia. Diferentes agregados fornecem diversos tipos de recursos, tais como de computação (máquinas virtuais). Alguns agregados fornecem soluções de rede para membros usarem na conexão de recursos de computação de vários agregados.

d) **API de gerenciamento de agregação:** membros podem solicitar recursos de agregados usando uma API padrão de gerenciamento de agregação. Ela permite as seguintes soluções:
   - Listar os recursos disponíveis em um agregado,
   - Solicitar recursos específicos se alocado às suas fatias,
   - Verificar a condição dos recursos que são atribuídos às suas fatias,
   - Excluir recursos das fatias.

A API usa documentos de especificação de recursos para descrevê-los, comumente referido como RSpec (*Resource Specification*). RSpecs são documentos XML (*eXtensible Markup Language*) apenas em um formato prescrito. Membros enviam para agregações um pedido RSpec que descreve os recursos solicitados e a agregação retorna um manifesto RSpec propondo os recursos disponíveis. No manifesto inclui informações dos recursos, tais como os nomes e endereços IP dos computadores (por exemplo, máquinas virtuais), contas de usuário e etiquetas de VLAN atribuídos aos *links* da rede. A maioria dos membros não precisam aprender os detalhes da API de gerenciamento de agregação ou escrever arquivos RSpec. O GENI possui algumas ferramentas que ocultam esta complexidade.

A Figura 3 mostra um membro criando uma fatia, recebendo credenciais do *Clearinghouse* (serviço da federação de confiança para todas as entidades de software: ferramentas, agregados, serviços) para acesso ao ambiente e adicionado recursos em um agregado usando a API de gerenciamento de agregação.

Figura 3 – Processo de experimentação na federação GENI



Fonte: Adaptado de GENI (2016).

Por fim, com os recursos de experimentação definidos, o pesquisador poderá instalar os seus *softwares*, executar, parar e coletar resultados do experimento.

2.2.1.2 Infraestrutura do GENI

O ponto fundamental de expansão do GENI é o campus implantar as seguintes tecnologias: *racks* GENI, OpenFlow/SDN e, possivelmente, WiMAX. Além disso, o campus deve concordar com etapas administrativas, incluindo tornar os recursos disponíveis aos investigadores locais e remotos em uma base contínua.

Atualmente projetos de *racks* podem ser caracterizados como segue:

a) **ExoGeni:** custo mais elevado, solução flexível, topologias de redes virtuais incluindo OpenFlow, que também fornece uma plataforma poderosa para aplicações em nuvem. Esses *racks* são normalmente implantados como parte integrante de uma rede de campus;
b) **InstaGENI:** custo de médio porte, expansível solução de *racks* que podem ter vários implantados em um campus, oferecendo suporte de aplicações em nuvem, juntamente com OpenFlow e redes VLAN. Esses *racks* são normalmente implantados fora de um firewall local;
c) **OpenGENI:** custo de médio porte, expansível solução de *racks*, proporcionando suporte de aplicações em nuvem, juntamente com OpenFlow e redes VLAN. Esses *racks* são implantados em hardware Dell;

d) **CiscoGENI:** solução dupla de *rack* que combina software ExoGeni com servidores das séries Cisco UCS-B e C. Esses *racks* apoiam OpenFlow, topologias virtuais e aplicativos de nuvem;
e) **CienaGENI**: solução de *rack* que combina software ExoGeni com *switches* Ciena. Esses *racks* ainda estão em desenvolvimento.

Todos os *racks* devem possuir conexões com a camada 3[6] para Internet convencional e ligações com a camada 2[7] para as redes principais do GENI (Internet2 AL2S). Os *racks* usam a Internet para controlar o acesso aos recursos e compartilhar VLANs para conexões de aplicativos e dados experimentais. Os *racks*, além disso, podem usar conexões de Internet da camada 3 para alguns experimentos, particularmente testes na nuvem.

O GENI pode ser visto como a confluência de vários grandes ambientes de teste, oferecendo alternativas para experimentação em redes. Dentre estes ambientes de teste pode-se destacar: PlanetLab, ORBIT e EmuLab (GENI, 2010) (Paul et al., 2011).

2.2.1.3 PlanetLab

O PlanetLab (PETERSON et al., 2006) é um amplo laboratório para testes e desenvolvimento de aplicações distribuídas, criado a partir de 2002 por um consórcio de instituições acadêmicas, governamentais e industriais, que montou uma grande malha de computadores espalhados pelo mundo em diversas redes. Atualmente, possui mais de 1050 máquinas espalhadas em mais de 550 locais diferentes, em todos os continentes. O projeto é dirigido a partir da Universidade de Princeton, EUA (*United States of America*), mas há segmentos, por exemplo, na Europa, onde há outros centros de desenvolvimento e controle dos recursos comuns.

O PlanetLab foi pioneiro no uso amplo dos conceitos de virtualização dos nós e de criação de fatias de recursos virtuais dedicados num experimento. No Brasil, a Rede Nacional de Ensino e Pesquisa (RNP) entrou como parceira do projeto em 2004, com a implantação de três nós nos pontos de presença (PoPs) da rede Ipê no Rio de Janeiro, no Ceará e no Rio Grande do Sul. Os seis equipamentos inicialmente instalados nos PoPs (dois para cada um) foram cedidos pela Intel. Mais recentemente, foram renovados os equipamentos nesses nós e foi acrescido um quarto nó no Pará.

[6] A camada 3 é a camada da Internet que possui o endereço IP (endereço lógico)
[7] A camada 2 é a camada de enlace que possui o endereço MAC da placa de rede (endereço físico)

No projeto GENI, o PlanetLab GENI (GENI, 2010), será uma alternativa de rede para experimentação aos seus pesquisadores. O grupo de Princeton dirige esta atividade e tem como escopo integrar logicamente os componentes GENI aos serviços PlanetLab como: PLC (*PlanetLab Central Software*), responsável por criar a rede sobreposta definindo a topologia virtual que será usada para o experimento; e o SFA (*Slice-Based Facility Architecture*), responsável por localizar e alocar os recursos para os experimentos (PETERSON et al., 2009).

2.2.1.4 ORBIT (*Open-Access Research Testbed for Next-Generation Wireless Networks*)

O projeto ORBIT provê uma rede sem fio flexível aberto à comunidade acadêmica para experimentação. Ele foi desenvolvido para que pesquisadores entendessem as limitações do mundo real das redes sem fio, que muitas vezes não são percebidas em simulações devido às simplificações aplicadas ao modelo ou por terem sido aplicadas em uma quantidade pequenas de nós.

O ORBIT possui dois ambientes de teste no campus da Universidade Rutgers: o primeiro é dentro de um laboratório, constituído por 400 nós dispostos em uma grade 20x20 (Figura 4), separados por um metro de distância entre os nós adjacentes; cada nó é composto por uma plataforma com múltiplas interfaces sem fio e com fio. O segundo está localizado ao ar livre, numa área de 1,5 hectares. Entre eles existem nós fixos, e ainda nós móveis para prover mobilidade entre os roteadores.

Figura 4 – Ambiente de teste no ORBIT

Fonte: Control and Management Framework – OMF (2011).

No GENI, sua integração permitirá que pesquisadores gerenciem esse ambiente de redes sem fio, além de estender nós do ORBIT para outros ambientes sem fio. No GENI, o ORBIT será integrado dentro do ambiente OMF (*cOntrol and Management Framework*) (Control and Management Framework – OMF, 2011).

2.2.1.5 EmuLab

EmuLab é um ambiente de teste em redes de computadores que oferece aos seus pesquisadores um leque de ambientes nos quais eles podem desenvolver, analisar e avaliar seus sistemas (EIDE et al., 2006). O nome EmuLab refere-se tanto ao ambiente de teste quanto ao *software* de interação do usuário com ele.

O projeto é gerenciado pela Universidade de Utah, onde foram desenvolvidos os primeiros nós do EmuLab (EMULAB, 2016). Atualmente, há mais de doze países utilizando o EmuLab, totalizando uma rede para experimentação com mais de cem nós. No Brasil, há um nó desta rede instalado na USP (Universidade do Estado de São Paulo).

A Figura 5 apresenta os clusters de computadores utilizados pelo EmuLab. Os ambientes disponíveis no EmuLab são: emulação - neste ambiente, o pesquisador define uma topologia arbitrária com *switches* ou roteadores e nós; livre-internet - onde o Emulab oferece um ambiente federado para experimentos sobre Internet; 802.11 Wireless - provê um ambiente com ponto de acesso, roteadores e cliente para experimentos em rede sem fio; e o ambiente radio definido por *software* que permite experimentações na camada 1 para análise de processamento de sinal.

Figura 5 – Ambiente de teste no EmuLab

Fonte: EmuLab (2016).

No GENI, a integração do EmuLab ao projeto deu origem a um subprojeto conhecido como ProtoGENI (PROTOGENI, 2011), e permitirá ao EmuLab expandir ainda mais seu ambiente de teste, além de controlar novos não oferecidos atualmente, por exemplo, a infraestrutura de altíssima velocidade da rede Internet.

2.2.2 FIRE (*Future Internet Research and Experimentation*)

A iniciativa FIRE na Europa visa a pesquisa experimental e ao financiamento de projetos que produzam infraestruturas para experimentação em Internet do Futuro. A meta é que as pesquisas em tecnologias para Internet do Futuro sejam direcionadas à rede ou a serviço e tenham a possibilidade de comparar as soluções correntes com as propostas futuras. Dessa forma, afirma-se que o FIRE possui duas dimensões relacionadas que direcionam suas pesquisas e seus investimentos (FIRE, 2008):

a) **Pesquisa experimental:** o objetivo é integrar a pesquisa multidisciplinar e a experimentação em larga escala. A partir daí, definir uma metodologia que direcionará a pesquisa experimental na infraestrutura FIRE, baseada em um ciclo interativo que vai dessa, passando pelo projeto e chegando à experimentação;
b) **Facilidades para testes:** o objetivo é oferecer múltiplos ambientes de teste, suportando várias tecnologias, interligadas e federadas entre si, para permitir a realização de experimentos envolvendo dois ou mais dos ambientes distintos. Desse modo, pretende-se que FIRE seja sustentável, renovável, dinâmico e integrado em larga escala. Deverá ainda facilitar a pesquisa experimental na comunidade acadêmica, nos centros de pesquisa e na indústria.

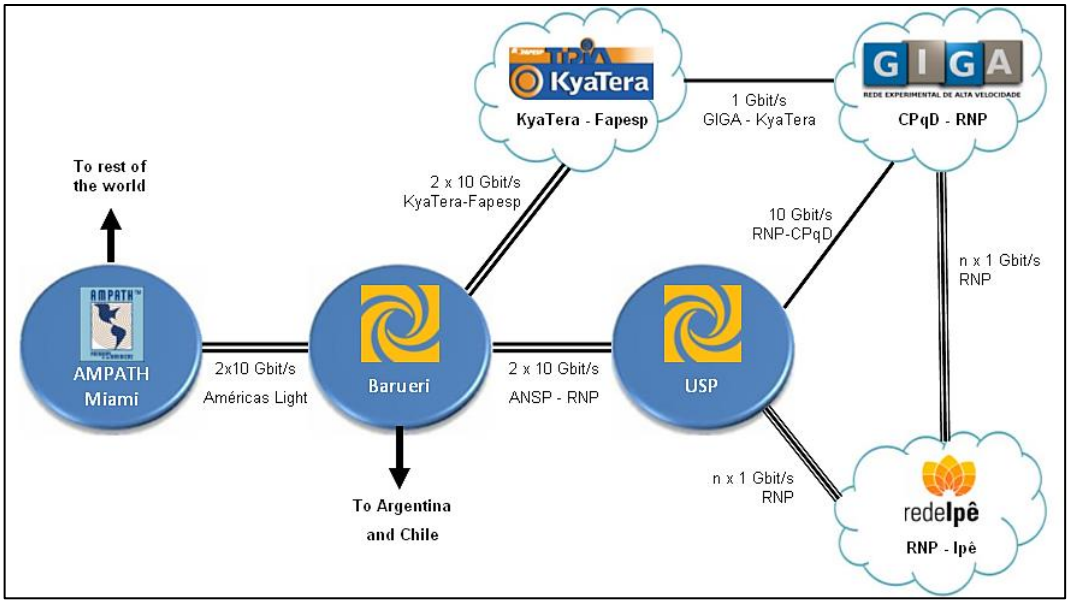
Para o FIRE, as experiências práticas são essenciais para dar credibilidade e levantar o nível de confiança na conclusão da pesquisa. Além disso, a experimentação deve ser executada em larga escala para que seja representativa, convincente e, para provar a escalabilidade da solução testada. Os projetos de ambiente de teste em destaque aqui financiados pela iniciativa FIRE incluem (FIRE, 2010): BonFIRE (*Building Service Testbeds on Future Internet Research and Experimentation*) (BONFIRE, 2013), CREW (*Cognitive Radio Experimentation World*) (CREW, 2009) e OFELIA (*OpenFlow in Europe: Linking Infrastructures and Applications*) (OFELIA, 2011).

2.2.3 Iniciativas brasileiras

Os parceiros brasileiros contribuem no projeto com a experiência na implantação de instalações de experimentações locais e na participação em diferentes projetos de pesquisa experimental em Internet do Futuro, embora com pouca ou nenhuma coalizão estratégica entre elas. O primeiro desses projetos a destacar-se é o de pesquisa e desenvolvimento GIGA e suas instalações experimentais em grande escala conhecido como rede GIGA (Rede Experimental de Alta Velocidade), coordenado conjuntamente pelo CPqD (Centro de Pesquisa e Desenvolvimento em Telecomunicações) e a RNP[8] (SCARABUCCI et al., 2005). O projeto GIGA atualmente se concentra em redes ópticas e as definidas por *software*.

A Rede GIGA possui um núcleo de 10 GbE e pontos de presença em sete cidades do sudeste brasileiro (Campinas, São Paulo, São José dos Campos, Cachoeira Paulista, Rio de Janeiro, Niterói e Petrópolis). Nestas localidades atende 26 instituições e dezenas de laboratórios, atualmente por meio de enlaces GbE. A Rede GIGA está interligada a outras redes experimentais sul-americanas, americanas e europeias através da rede Ipê, viabilizando a experimentação e validação de tecnologias de TIC em escala planetária (GIGA, 2011). A Figura 6 apresenta a conectividade da Rede GIGA com outras redes.

Figura 6 – Conectividade da rede GIGA com outras redes



Fonte: GIGA (2011).

[8] Rede Ipê - rede de produção e pesquisa que interliga as universidades federais do Brasil.

O segundo projeto é o Web Science (Pesquisa em Ciência da Web) (WEBSCIENCE, 2010), apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) dentro do programa Institutos Nacionais de Ciência e Tecnologia (INCT). O projeto Web Science começou efetivamente em 2010 e o subprojeto de Arquiteturas para a Internet do Futuro envolve a RNP, dez universidades brasileiras parceiras: PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro), UFRJ (Universidade Federal do Rio de Janeiro), UNICAMP (Universidade Estadual de Campinas), UNIRIO (Universidade Federal do Estado do Rio de Janeiro), UFF (Universidade Federal Fluminense), UERJ (Universidade do Estado do Rio de Janeiro), UENF (Universidade Estadual do Norte Fluminense), UFC (Universidade Federal do Ceará), UFRN (Universidade Federal do Rio Grande do Norte), LNCC (Laboratório Nacional de Computação Científica) e cinco universidades internacionais: DERI (*Digital Enterprise Research Institute*), L3S (L3S *Research Center*), LIP6 (*Laboratoire d'Informatique de Paris VI*), LERO (*The Irish Software Engineering Research Centre*) e U.Waterloo (*University of Waterloo*), com experiência em redes ópticas e sem fio, estudos de simulação e emulação e monitoramento de rede. Um dos primeiros objetivos do projeto é estabelecer ilhas experimentais nos laboratórios de cada parceiro e interligá-las na camada 2 por meio das redes Ipê e GIGA.

O terceiro projeto em destaque é o FIBRE (*Future Internet Testbeds Experimentation between Brazil and Europe*) (FIBRE, 2010). Este projeto é financiado de forma conjunta pelo CNPq e pela Comunidade Europeia, visando a criação e expansão de uma infraestrutura de rede compartilhada entre Brasil e Europa para apoiar a realização conjunta de pesquisas experimentais no âmbito da Internet do Futuro, em escalas e condições mais próximas da realidade. Atualmente, a infraestrutura FIBRE possui alguns pontos de presença no Brasil: RNP em Brasília-DF, UFPE (Universidade Federal de Pernambuco) em Recife-PE, UFRJ no Rio de Janeiro-RJ, UFF em Niterói-RJ, UNIFACS (Universidade Salvador - *Laureate International Universities*) em Salvador-BA, UFG (Universidade Federal de Goiás) em Goiânia-GO, UFPA (Universidade Federal do Pará) em Belém-PA, UFRGS (Universidade Federal do Rio Grande do Sul) em Porto Alegre-RS, UFES (Universidade Federal do Espírito Santo) em Vitória-ES, USP em São Paulo-SP, UFSCar (Universidade Federal de São Carlos) em São Carlos-SP, CPqD em Campinas-SP. O CPqD é um dos principais articuladores do consórcio FIBRE e colabora tecnicamente com o projeto, provendo uma ilha com recursos de rede que poderão ser interligados a ilhas de outras instituições. O objetivo é estabelecer uma conexão entre elas, criando uma experiência laboratorial com equipamentos comerciais, em maior escala e com condições de rede semelhares às redes operacionais.

3 VIRTUALIZAÇÃO

Nesta seção, será apresentada uma introdução sobre virtualização, seguida de uma breve descrição sobre virtualização de sistemas para entender como funciona a virtualização de redes (FARIAS et al., 2011).

A virtualização é uma técnica que permite que um sistema execute processos sobre recursos dedicados. Inicialmente, como um mecanismo de isolamento, ela representa um fator de uso eficiente da crescente capacidade computacional disponível (EGI et al., 2010) e integra arquiteturas com elementos comuns a um conjunto de processos virtualizados que possuem apenas uma cópia em execução, acessada de forma compartilhada (BHATIA et al., 2008).

O conceito foi estendido do âmbito de nós para os demais elementos de uma rede de computadores, dando origem à virtualização de redes (CHOWDHURY; BOUTABA, 2010).

Em um processo paralelo àquele descrito para a virtualização tradicional, a aplicação da virtualização de redes passou a permitir que os componentes de uma rede física particionassem sua capacidade de maneira a realizar simultaneamente múltiplas funções, estabelecendo infraestruturas lógicas distintas e mutuamente isoladas.

Assim, como no caso da virtualização de sistemas, a virtualização de redes, do mesmo modo, permitiu que as arquiteturas de redes se tornassem mais eficientes. Funções que tradicionalmente eram gerenciadas de forma distribuída passaram a ser projetadas para uma execução e administração centralizadas. É o caso do encaminhamento do tráfego IP: podem ser encontradas arquiteturas do estado da arte (NASCIMENTO et al., 2010) em que decisões de roteamento, originalmente tomadas de forma local por nós especializados, são encaminhados por comutadores a um sistema controlador, que executa em memória uma versão virtualizada da rede e dos respectivos elementos roteadores. Deriva decisões da base de informações de roteamento construída pela execução desta rede virtual e as transmite aos comutadores, que reagem de acordo.

3.1 VIRTUALIZAÇÃO DE REDES

As novas ideias e implementações no núcleo da rede tolera discriminação pelos administradores de rede devido à falta de confiança no bom funcionamento na rede, devido aos riscos de indisponibilizar a rede, além do custo benefício a eles envolvidos. Uma das propostas vistas como alternativa para o desenvolvimento de inovações juntamente ao tráfego de produção dar-se com a virtualização de redes (COSTA, 2013).

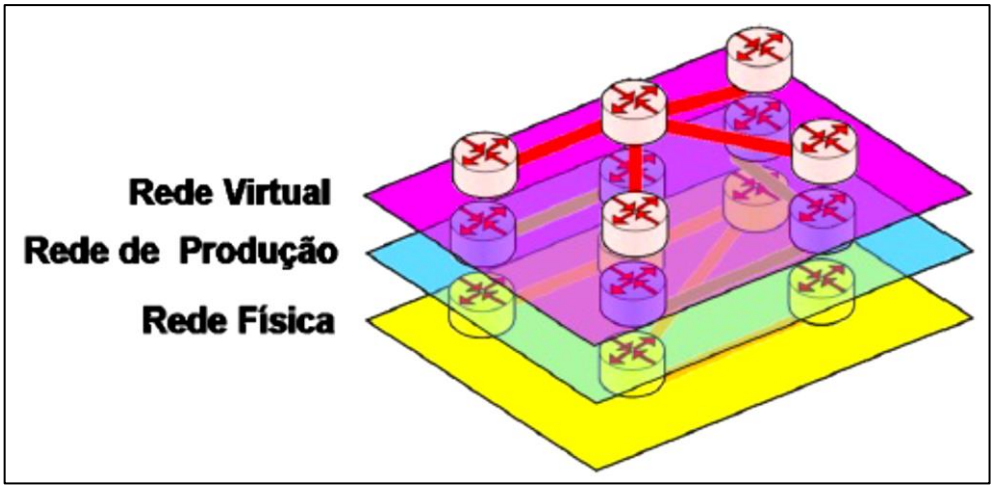
Para explicar a virtualização de redes, primeiramente vale conceituar a virtualização de sistemas. A virtualização de sistemas é uma técnica que permite que um nó computacional, ou seja, uma máquina execute múltiplos processos oferecendo a cada um deles a aparência de estar executando sobre recursos dedicados. Dessa forma, cada máquina virtual acessa interfaces similares à máquina real. Para que as máquinas virtuais estejam sendo executadas sobre uma máquina real, os ambientes virtuais devem ser isolados, ou seja, a execução de uma máquina virtual não deve interferir na outra, assim uma máquina virtual deve ter acesso a máquina real como se fosse a única. Nesse contexto, a virtualização de redes igualmente faz uma abstração de um recurso.

A virtualização de redes passou a permitir que os componentes de uma rede física compartilhassem sua capacidade de maneira que realize simultaneamente múltiplas funções, estabelecendo infraestruturas lógicas distintas e mutuamente isoladas, provendo um método para que múltiplas arquiteturas de rede distintas compartilhem o mesmo substrato físico. Em outras palavras, uma rede virtual é uma rede composta pela interconexão de um conjunto de roteadores virtuais que representam uma “fatia” de roteadores físicos compartilhados.

Como as redes virtuais são isoladas, o tráfego experimental não afetará o trafego de produção (Figura 7). Esse modelo consiste em uma das principais abordagens para as redes de computadores do futuro, em que cada rede virtual é isolada e possui sua própria pilha de protocolos e seu gerenciamento individual.

A topologia de uma rede virtual não precisa ser idêntica à topologia de uma rede física, embora seja necessária uma rede física para transportar os dados. Uma rede virtual é uma rede composta por roteadores lógicos conectados em uma determinada topologia. Dessa maneira, os enlaces virtuais são criados pelo particionamento dos enlaces físicos, tal particionamento corresponde a uma “fatia” da banda disponível no enlace físico, portanto, a banda do enlace físico é dividida entre os enlaces virtuais.

Figura 7 – Rede virtualizada, com três redes compartilhando o mesmo substrato físico



Fonte: Mattos (2011).

Como a representação da topologia das redes virtuais não é necessariamente igual a topologia da rede real, para representar roteadores virtuais adjacentes em roteadores físicos não adjacentes, utiliza-se o tunelamento, que cria túneis entre os roteadores físicos para tornar a rota de transmissão transparente para a topologia virtual. Essa característica de poder mapear a rede virtual em uma rede física permite que as redes virtuais sejam extremamente flexíveis. Para que haja essa flexibilidade faz-se necessário uma função de migração de redes.

Uma função de controle fundamental nas redes virtualizadas é a migração de redes virtuais. A migração faz a movimentação dos nós das redes virtuais sobre os nós da rede física (WANG et al., 2008). Pode-se aplicar a migração de diversas maneiras como, por exemplo, a manutenção de nós na rede, que algumas vezes causam uma quebra na conexão, devido ao seu desligamento causando um atraso para que as rotas sejam reorganizadas entre os nós. Assim, com a migração, pode-se contornar esse cenário com uma simples migração do nó virtual para um nó físico que esteja em funcionamento. Dessa maneira, as redes virtuais garantem que a topologia lógica não seja alterada e assim as rotas continuem válidas.

Além dos casos de manutenção podem-se destacar os casos de ataques de negação de serviço, bastando trocar todos os nós do substrato em ataque para outros nós físicos fora da região do ataque; casos para economia de energia, quando há nós físicos subutilizados à migração pode ser detectada na rede física e assim, serem desligados entre outros.

A migração de redes virtuais ainda sofre alguns desafios tais para remapear os enlaces virtuais sobre um ou mais enlaces físicos e como reduzir o tempo de migração, visto que os nós virtuais ficam indisponíveis durante o processo possibilitando a perda de pacotes no núcleo da rede. As redes virtuais trazem novos horizontes para o núcleo da rede. Com essa ampla

flexibilidade, as redes virtuais permitem a instanciação, a remoção e a configuração de recursos de redes virtuais, sob demanda, permitindo ainda que as redes sejam monitoradas enquanto estão ativas. Dessa maneira, a virtualização tem sido amplamente usada para o desenvolvimento de propostas para a Internet do Futuro (MOREIRA et al., 2009), e para o desenvolvimento de redes experimentais. Assim, a comunidade científica começou a investir mais no tema, já que as redes virtuais são capazes de oferecer ambientes reais para testes de novas ideias.

Nesse contexto, os pesquisadores de rede estão trabalhando em algumas redes virtuais programáveis como GENI (GENI, 2016) citado anteriormente, como um centro de investigação para a experimentação de novas arquiteturas de rede e sistemas multiusuários. Esse tipo de rede tem como objetivo possibilitar a experimentação de protocolos em larga escala por meio de *switches* e roteadores programáveis utilizando virtualização. Para tal, essa rede deve possuir *switches* e roteadores espalhados mundialmente e podem ser modificados para executar protocolos experimentais. A proposta de GENI, em geral, é oferecer ao pesquisador uma "fatia" dos recursos da rede que consiste em enlaces, roteadores, *switches* e terminais para que o desenvolvedor possa executar seus experimentos, por meio da virtualização, podendo configurar seus recursos como desejar.

As redes virtualizadas, embora forneçam uma ampla flexibilidade para controlar o núcleo da rede, carregam consigo algumas desvantagens. O gerenciamento de uma rede virtual se parece muito com o gerenciamento de uma rede física, além disso, tem-se uma banda muito limitada, visto que a banda de enlace virtual é uma fatia da banda física real. Outro problema é o mapeamento dos nós virtuais entre os reais, que exige ampla atenção, uma vez que se um dos nós físicos parar de funcionar, todos os nós virtuais a ele referenciados deixarão de funcionar até que haja uma migração, causando prejuízos e atrasos na transmissão da rede.

Nesse cenário, o paradigma de redes virtuais é parte de uma das abordagens e propostas para a Internet do Futuro.

4 OPENFLOW

Observa-se nas seções anteriores que as redes de computadores atuais precisam de aprimoramento e para isso seguem algumas propostas para a Internet do Futuro. Uma das concepções promissoras em destaque no meio acadêmico são as Redes Definidas por Software.

Nesse contexto, as redes SDN trazem consigo a capacidade de controlar o plano de encaminhamento dos pacotes da rede, no entanto para que isso ocorra é necessária uma interface que seja simples para realizar a comunicação entre os elementos de comutação da rede e o controlador da rede. De fato, a interface que foi associada desde o início ao paradigma SDN, sendo um dos elementos mais motivadores para sua criação, foi o OpenFlow.

O OpenFlow foi proposto pela Universidade de Stanford e seu objetivo inicial é atender à demanda de validação de novas propostas de arquitetura e protocolos de rede sobre equipamentos comerciais. Dessa forma, é possível implementar uma tecnologia capaz de promover a inovação no núcleo da rede, por meio da execução de redes de teste em paralelo com as redes de produção (MCKEOWN, 2008).

A proposta do OpenFlow promove a criação de redes SDN, usando elementos comuns de rede, tais como *switches*, roteadores, pontos de acesso ou, até mesmo, computadores pessoais.

Esta seção apresenta as características do padrão OpenFlow, bem como os elementos necessários para sua configuração e aplicação em uma Rede Definida por Software. Além disso, é apresentado o comutador OpenFlow, suas características, seu funcionamento e como é utilizado entre os comutadores OpenFlow e um controlador SDN.

4.1 PADRÃO OPENFLOW

O OpenFlow é um padrão aberto para o paradigma de redes SDN. Ele consiste de uma interface de programação que permite ao desenvolvedor controlar diretamente os elementos de encaminhamento de pacotes presentes no dispositivo.

A pesquisa na área de redes de computadores possui diversos desafios em relação à implementação e à experimentação de novas propostas de rede em ambientes reais. O pesquisador de redes em geral não possui uma rede de teste que apresente desempenho próximo de uma rede real. Para isso foi proposto o OpenFlow permitindo que um pesquisador possa experimentar suas propostas em redes reais, sem atrapalhar o fluxo de produção, obtendo uma melhor validação na sua pesquisa para a indústria.

Dessa forma, os pesquisadores podem reprogramar os comutadores de modo que não interfira no tráfego de produção, ou seja, a rede continuará funcionando do mesmo modo.

O projeto do OpenFlow tem por finalidade ser flexível e atender os seguintes requisitos estipulados:

a) Capacitar o uso em implementação de baixo custo e alto desempenho;
b) Possibilitar o suporte de uma vasta gama de pesquisas científicas;
c) Garantir o isolamento do tráfego de produção e o tráfego experimental;
d) Garantir que não seja necessário a exposição do projeto dos fabricantes em suas plataformas.

Uma das características básicas do padrão OpenFlow é a separação clara entre os planos de dados e de controle nos elementos de comutação. O plano de dados é responsável pelo encaminhamento dos pacotes associando entradas na tabela de encaminhamento de cada comutador; essas regras são definidas pelo padrão e incluem os seguintes itens como tomada de ação:

a) Encaminhar o pacote para uma porta especifica do dispositivo;
b) Alterar parte de seus cabeçalhos;
c) Descartar o pacote;
d) Encaminhar o pacote para a análise de um controlador de rede.

É necessário deixar claro que o plano de dados pode ser implementado em *hardware* utilizando elementos comuns a roteadores e *switches* atuais. Em geral, essa estrutura se configura em dispositivos dedicados que tenha a interface de configuração OpenFlow, por exemplo, comutadores de uma rede de produção que se queira fazer testes experimentais de pesquisas de rede.

Por outro lado, o plano de controle fica responsável pelo controle da rede permitindo ao controlador SDN da rede programar as entradas dessa tabela de encaminhamento com padrões que identificam fluxos de interesse e as regras associadas a eles. O elemento responsável pelo plano de controle pode ser um módulo em *software* implementado de forma independente em algum ponto da rede. Esse elemento, denominado controlador SDN, fica responsável por instalar regras e ações no *hardware* de rede.

4.2 COMPONENTES DE UMA REDE OPENFLOW

Para que uma rede programável com OpenFlow exista é necessário que sua arquitetura seja composta por quatro componentes a ela associados. São necessários equipamentos habilitados que tenham a capacidade de alterar suas tabelas de encaminhamento, conforme as decisões de um controlador em software a eles conectados, por meio de um canal seguro (ROTHENBERG et al., 2010).

a) **Tabela de fluxos:** cada entrada na tabela de fluxos do *hardware* de rede é composta por regras, ações e controles de estatística. A regra é formada pela definição dos valores de um ou mais campos do cabeçalho do pacote, e é por meio dela que é determinado o fluxo. As ações, então, ficam associadas ao fluxo e vão determinar como os pacotes devem ser processados, para onde vão ser encaminhados ou se serão descartados. Os controles de estatística consistem de contadores utilizados para manter estatísticas de utilização e para remover fluxos inativos ou que não existam mais. Logo, as entradas da tabela de fluxos são interpretadas pelo *hardware* como decisões em cache do plano de controle em *software*, sendo, portanto, a mínima unidade de informação no plano de dados da rede;

b) **Protocolo OpenFlow**: é o protocolo para a comunicação entre o comutador OpenFlow e o controlador de rede para que haja a troca de mensagens por um canal seguro. Essas mensagens podem ser simétricas, assíncronas ou ainda iniciadas pelo controlador;

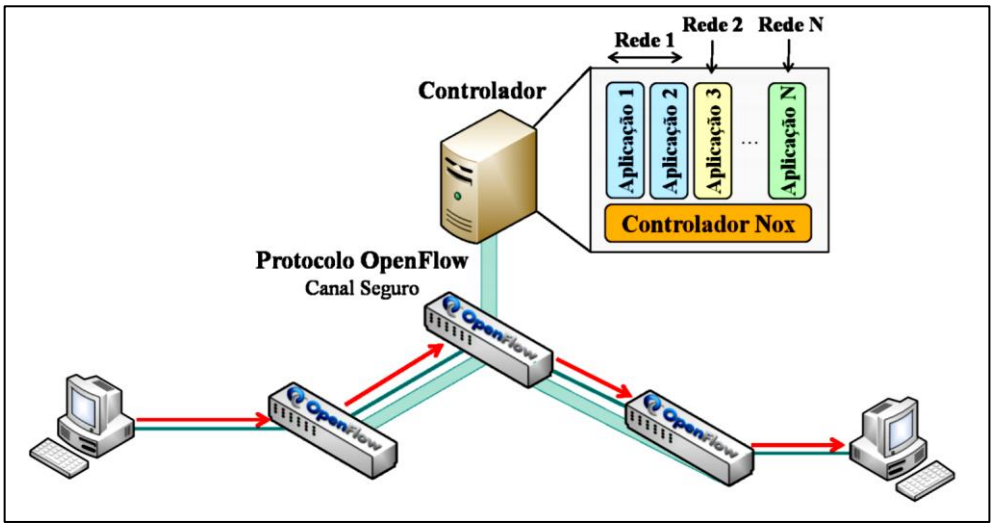
c) **Controlador:** o controlador fica responsável por tomar as decisões adicionando e removendo entradas na tabela de fluxos de acordo com uma política de encaminhamento. O controlador é uma camada de abstração da infraestrutura física que tem como objetivo facilitar o desenvolvimento de aplicações e serviços que gerenciam as entradas de fluxo na rede. Esse modelo de abstração se assemelha aos outros sistemas de software que provêm a abstração do *hardware* como, por exemplo, os sistemas operacionais de computadores pessoais. Ele permite a evolução em paralelo das tecnologias no plano de dados e as inovações na lógica das aplicações de controle;

d) **Canal seguro:** esse elemento não só garante que uma rede SDN não sofra ataques de elementos mal-intencionados como garante, além disso, que a troca de informações entre os comutadores e controladores da rede sejam confiáveis, com baixa taxa de erros. A interface de acesso que o projeto do OpenFlow recomenda é

o SSL (*Secure Socket Layer*), no entanto, existem alternativas como o TCP e o PCAP (*Packet Capture*) sendo muito úteis em ambiente virtuais e de experimentação por sua simplicidade de utilização.

A Figura 8 mostra a arquitetura de uma rede OpenFlow. Os comutadores OpenFlow se comunicam com o controlador por meio do protocolo OpenFlow em um canal seguro. O controlador adiciona ou remove entradas nas tabelas de fluxos dos comutadores de acordo com as aplicações de controle de cada rede virtual.

Figura 8 – Componentes de uma rede OpenFlow



Fonte: Mattos (2011).

4.3 PROTOCOLO OPENFLOW

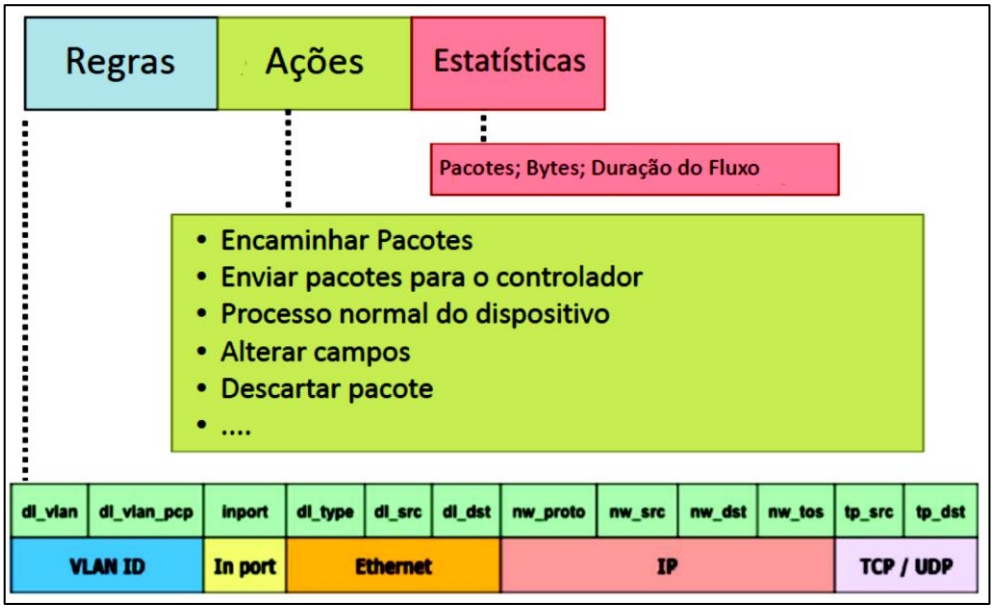
Uma das vantagens em se utilizar uma arquitetura OpenFlow é a flexibilidade que ela oferece para se programar de forma independente o tratamento de cada fluxo da rede e como ele deve ou não ser encaminhado nela. De uma maneira mais prática, o OpenFlow determina como um fluxo pode ser definido, quais serão as ações que podem ser realizadas para cada pacote pertencente a este fluxo e qual é o protocolo de comunicação entre o controlador e os comutadores utilizados para realizar as definições de fluxo e ações.

Dessa forma, uma entrada na tabela de fluxos de um comutador OpenFlow é formada pela união da definição do fluxo e um conjunto de ações a ele determinado.

Um fluxo é constituído pela definição dos valores de um ou mais campos do cabeçalho do pacote a ser processado pelo dispositivo OpenFlow. Por esse processo, os campos do cabeçalho descrevem o fluxo que, por sua vez, descrevem quais pacotes combinam com aquele

fluxo. Esses campos formam uma dupla de doze elementos que reúnem características dos protocolos da camada de enlace, de rede, e de transporte segundo o modelo TCP/IP (*Transmission Control Protocol*/*Internet Protocol*). Cada fluxo contém regras para a definição dos pacotes, ações e contadores estatísticos a ele associados, como pode ser visto na Figura 9.

Figura 9 – Definição de um fluxo na arquitetura OpenFlow



Fonte: Adaptado de Mckeown (2008).

Com a definição de fluxo, é possível observar que as regras de encaminhamento de um pacote não se restringem ao endereço IP ou endereço MAC dos pacotes. O protocolo OpenFlow é mais abrangente de maneira que os elementos formados pelos campos do pacote, as duplas, podem conter valores exatos ou valores arbitrários, que combinam com qualquer valor que o pacote comparado apresente para o campo. O encaminhamento pode se dar por outras características do pacote, como por exemplo, as portas de origem e destino do protocolo de transporte. Essas características descrevem o funcionamento do protocolo OpenFlow explicitados adiante.

4.4 CONTROLADOR OPENFLOW

O controlador OpenFlow consiste em um controle centralizado, seja fisicamente ou logicamente, que executa aplicações de controle sobre a rede OpenFlow configurando e gerenciando as tabelas de fluxo dos elementos encaminhadores.

É obviamente necessário que o controlador implemente o protocolo OpenFlow para se comunicar, por meio de mensagens, com os elementos encaminhadores (comutadores com OpenFlow habilitado), enviando os comandos para a rede. O controlador age como um sistema operacional de rede agindo sobre o plano de controle como uma interface entre as aplicações de controle da rede e a própria rede.

As mensagens enviadas pelo controlador podem ou não necessitar de resposta do elemento encaminhador e são classificadas de acordo com os seguintes tipos:

a) **Mensagens de características:** geralmente ocorrida após o estabelecimento do canal entre o comutador e o controlador OpenFlow. O controlador solicita ao comutador o pedido de conexão, bem como as características específicas e as capacidades do mesmo para que o controlador possa gerenciar melhor a rede;
b) **Mensagens de configurações:** o controlador é capaz de definir e consultar parâmetros de configuração no comutador. O comutador responde à solicitação de consulta do controlador;
c) **Mensagens de modificações do estado:** são mensagens enviadas pelo controlador para gerenciar os estados dos comutadores. Seu principal objetivo é adicionar ou remover entradas nas tabelas de fluxo dos comutadores, ou seja, adicionar e remover regras;
d) **Mensagens de leitura do estado**: são mensagens enviadas pelo controlador para ler o estado e coletar as estatísticas (contadores) do controlador;
e) **Mensagens de *packet-out***: essas mensagens são usadas para enviar pacotes completos para o comutador. Quando não há uma correspondência entre o pacote e alguma entrada na tabela de fluxos do comutador, o pacote então é encaminhado, por completo, para o controlador (*packet-in*). Esse, por sua vez, reenvia, utilizando o *packet-out*, o mesmo pacote com seu conjunto de ações a ele associado;
f) **Mensagens de barreira:** são mensagens utilizadas pelo controlador para garantir se as dependências de mensagens foram cumpridas ou para receber notificações de operações concluídas.

4.5 FUNCIONAMENTO

Definido o protocolo e o controlador OpenFlow, o funcionamento de uma rede OpenFlow é programado a partir do plano de controle. Os fluxos podem ser definidos da forma escolhida pelo controlador, como por exemplo, todos os pacotes enviados pela máquina com IP

“X” para a máquina com IP “Y” ou, ainda, todos os pacotes pertencentes à VLAN (*Virtual Local Area Network*) “Z”. As ações são definidas a partir destes fluxos e incluem:

a) Encaminhar os pacotes pertencentes àquele fluxo para determinada (s) porta (s);
b) Enviar os pacotes para o controlador;
c) Descartar o pacote como medida de segurança;
d) Modificar os campos dos cabeçalhos dos pacotes;
e) Encaminhar o pacote para o processamento normal do dispositivo.

As ações do último tipo (e) garantem que o tráfego experimental não interfere no tráfego de produção da rede. Outra maneira de garantir isso é definindo um conjunto de VLANs para fins experimentais. Com essa separação dos tráfegos é possível ter equipamentos híbridos que processam tráfego legado de acordo com os protocolos e as funcionalidades embarcadas do equipamento e, ao mesmo tempo, de maneira isolada, obtém o tráfego baseado nas regras do OpenFlow. As ações associadas aos fluxos formam as entradas nas tabelas de fluxo do comutador OpenFlow.

Além das ações, a arquitetura OpenFlow também possui contadores para o controle de cada fluxo. Cada fluxo contém contadores para o controle de número de pacotes; números de bytes trafegados no fluxo; e duração do fluxo. Esses contadores são implementados para cada entrada da tabela de fluxos e podem ser controlados e acessados pelo controlador por meio do protocolo OpenFlow.

Após definida a entrada na tabela de fluxo, cada pacote que chega a um comutador OpenFlow é comparado com cada entrada nessa tabela e caso seja encontrado, considera-se que o pacote pertence àquele fluxo e por sua vez irá agir de acordo com uma ação preestabelecida, como exemplo: ações do tipo (a) ou (d). Caso não seja encontrado, o pacote é encaminhado por completo para o controlador para ser processado, ação do tipo (b), ou pode ser apagado de acordo com alguma política de segurança preestabelecida, ação do tipo (c). Alternativamente, apenas o cabeçalho é encaminhado ao controlador mantendo o pacote armazenado no buffer do *hardware*. Caso o pacote seja encaminhado ao controlador, ele analisará o pacote e dependendo da análise da aplicação poderá criar uma nova entrada para aquele fluxo.

Em geral, os pacotes que chegam ao controlador sempre correspondem ao primeiro pacote de um novo fluxo ou, dependendo da aplicação, o controlador pode optar por instalar uma regra no comutador para que todos os pacotes de determinado fluxo sejam enviados para o controlador para serem tratados de maneira individual. Esse último caso corresponde normalmente a pacotes de controle: ICMP (*Internet Control Message Protocol*), DNS (*Domain*

Name System) e DHCP (*Dynamic Host Configuration Protocol*) ou protocolos de roteamento: OSPF (*Open Shortest Path First*) e BGP (*Border Gateway Protocol*) (ROTHENBERG, 2010).

Essas tabelas de fluxo criam um conjunto de regras que podem servir em diversas possibilidades, muitas das funcionalidades que são implementadas separadamente podem ser agrupadas em um único controlador OpenFlow, como por exemplo, funcionalidades que implementam serviços como NAT (*Network Address Translation*), firewall e VLANs. Alguns exemplos de possibilidades são apresentados na Tabela 1, em que cada linha representa um fluxo com suas respectivas ações a serem tomadas. O campo representado por um "*" indica que qualquer valor é aceito naquela posição, ou seja, é um campo que não importa no reconhecimento do fluxo (MCKEOWN, 2008).

Tabela 1 – Exemplo de uma tabela de fluxos do comutador OpenFlow

Port	**VLAN**	**ETH SRC**	**ETH DST**	**IP SRC**	**IP DST**	**IP Proto**	**L4 SRC**	**L4 DST**	**Ações**
*	*	*	00:4F:...	*	*	*	*	*	Port 4
*	*	*	*	*	*	TCP	*	22	DROP
*	1	*	00:4F:...	*	*	*	*	*	Port 4,6,9

Fonte: Mckeown (2008).

Apesar de o protocolo OpenFlow possuir um pequeno conjunto de ações simples, ele ainda é capaz de prover uma vasta gama de possibilidades para o desenvolvimento de aplicações de rede. Isso se deve ao fato que esses conjuntos de ações, se bem combinados e desenvolvidos em uma aplicação, podem atender aos requisitos específicos de um determinado fluxo. Somando-se isso ao fato de uma rede OpenFlow executar múltiplas aplicações em paralelo, atendendo os requisitos específicos de diversos fluxos com um controle lógico central, o OpenFlow torna-se uma excelente alternativa para a nova arquitetura da Internet do Futuro.

4.6 APLICAÇÕES DO OPENFLOW

O OpenFlow permite o experimento de novas propostas na área de redes de computadores podendo até mesmo utilizar uma infraestrutura de rede já existente. Dentre as possíveis aplicações que se pode realizar com uma rede OpenFlow, podem ser classificadas as seguintes (MCKEOWN, 2008):

a) **Novos protocolos de roteamento:** aplicações diversas podem tratar de fluxos específicos de uma rede OpenFlow. Assim quando um pacote de um novo fluxo chegar a um comutador, ele é encaminhado para o controlador que é responsável por escolher a melhor rota para o pacote seguir na rede a partir de uma política adotada pela aplicação (um novo protocolo de roteamento) correspondente. Após isso, o controlador adiciona as entradas na tabela de fluxos de cada comutador pertencente à rota escolhida e os próximos pacotes desse fluxo que forem encaminhados na rede não necessitarão ser enviados para o controlador. Dessa forma, novos protocolos de roteamento podem ser implementados para diversos fluxos específicos na rede de modo que tais protocolos cuidem separadamente de um conjunto de fluxos;
b) **Mobilidade:** uma rede OpenFlow que possui pontos de acesso sem fio permite que clientes móveis utilizem sua infraestrutura para se conectarem à Internet. Assim, mecanismos de *handoff* podem ser executados no controlador com uma simples migração dinâmica das tabelas de fluxo dos comutadores de acordo com a movimentação do cliente, permitindo a redefinição da rota utilizada;
c) **Redes não IP**: o comutador OpenFlow analisa arbitrariamente os campos do pacote para definir a qual fluxo ele pertence, de acordo com a tabela de fluxos do comutador, permitindo a flexibilidade na definição deste. A partir de então, podem ser testadas, por exemplo, novas formadas de endereçamento e encaminhamento para as redes de computadores sob a arquitetura OpenFlow;
d) **Redes com processamento de pacotes**: existem aplicações OpenFlow que realizam processamento de cada pacote de um fluxo; nesse caso, a aplicação implementada no controlador OpenFlow cria uma regra nos comutadores da rede forçando que cada pacote recebido seja enviado ao controlador para ser analisado e processado pela aplicação. Como já mencionado, esses casos correspondem, em geral, a pacotes de controle (ICMP, DNS, DHCP) ou protocolos de roteamento (OSPF, BGP).

4.7 O OPENFLOW ATUALMENTE

O OpenFlow pode ser visto como uma tecnologia promissora e inovadora que abre uma série de oportunidades de desenvolvimento tecnológico na área de redes de computadores. Esse fator potencial da tecnologia OpenFlow tem atraído a atenção da indústria no desenvolvimento de protótipos com suporte ao OpenFlow, tais como: HP (Hewlett-Packard), NEC (Nippon Electric Company), Extreme, Arista, Ciena, Juniper, Cisco e Huawei; no suporte dos

fornecedores de processadores em silício (Broadcom, Marven); na criação de empresas como a Nicira e a Big Switch Networks; e no interesse de operadoras e provedores de serviço, tais como Deutsche Telekom, Docomo, Google, Facebook e Amazon. Espera-se, à medida que a especificação OpenFlow evolua, mais fabricantes de equipamentos incorporem o padrão em suas soluções comerciais.

O projeto, inicialmente desenvolvido na Universidade de Stanford, continua sendo desenvolvido pela *Open Networking Foundation* (ONF). A primeira versão foi a 0.2.0 lançada em maio de 2008 e atualmente está obsoleta. A versão 1.0.0 lançada em dezembro de 2009 foi a mais amplamente divulgada e implementada. Após a versão 1.0.0, foram lançadas as versões 1.1, 1.2, e 1.3 e 1.4.

Em maio de 2012 a Universidade de Indiana em parceria com a ONF lançou o SDN *Interoperability Lab* que incentiva o desenvolvimento e a adoção de normas paras as redes SDN com a tecnologia OpenFlow. Em fevereiro de 2012, a Big Switch Networks lançou o controlador Floodlight. Nesta mesma data, a HP anunciou que estava tomando providências para lançar os seus primeiros equipamentos de rede com OpenFlow habilitado. Em abril de 2012, a Google descreveu como uma rede interna da empresa tinha sido reprojetada para uma arquitetura OpenFlow com melhoria de desempenho e eficiência relevantes.

Assim, demonstra-se que a consolidação das tecnologias de equipamentos programáveis em *software* no padrão OpenFlow, ampliou o conceito de Redes Definidas por Software envolvendo-o a novos paradigmas de gerência integrada, inovação em protocolos e serviços baseados em recursos de redes virtualizadas ou definidas por software.

5 REDES DEFINIDAS POR SOFTWARE

Nesta seção será apresentada a arquitetura, os elementos programáveis e os controladores das Redes Definidas por *Software*, bem como aplicações que abordam os principais contextos que podem beneficiar com essa estrutura.

Observa-se, apesar de toda evolução das redes de computadores, em termos de aplicações, sua arquitetura, não evoluiu da mesma forma e por esse motivo não contempla significamente todos os requisitos atuais da Internet. Além do mais, a Internet tornou-se comercial e, por sua vez, seus equipamentos de rede se tornaram caixas pretas, isto é, implementações baseadas em *software* fechado sobre um hardware próprio (COSTA, 2013).

Contrapondo a abordagem da arquitetura atual das redes de computadores, surgiram ideias e pesquisas sobre a Internet do Futuro, pesquisas que visam resolver os problemas até então enfrentados pela sua arquitetura monolítica distributivamente fechada, tais como problemas associados ao endereçamento, mobilidade, escalabilidade, gerenciamento e qualidade de serviço.

As abordagens da Internet do Futuro visam resolver os problemas atuais e atender às novas demandas de requisitos estabelecidos pela Internet. Em geral, essas abordagens caracterizam-se por um plano de controle concentrado que permitem mover grande parte da lógica de tomada de decisões dos dispositivos de redes para controladores externos, que podem ser implementados com o uso de tecnologia de servidores comerciais, um recurso abundante, escalável e barato (ROTHENBERG et al., 2010).

Nos equipamentos de redes tradicionais, as tomadas de decisão ocorrem no seu próprio interior, ou seja, o roteamento dos pacotes de rede é definido por algoritmos previamente calculados, geralmente fechados, de difícil ou impossível modificação; uma vez implementado em uma rede o equipamento, adota suas próprias decisões para enviar o pacote.

Se o controle das tomadas de decisão fosse logicamente centralizado, haveria a possibilidade da definição do comportamento da rede em software, não apenas pelos próprios fabricantes do equipamento, mas ainda por fornecedores ou pelos próprios usuários, como, por exemplo, operadores de rede.

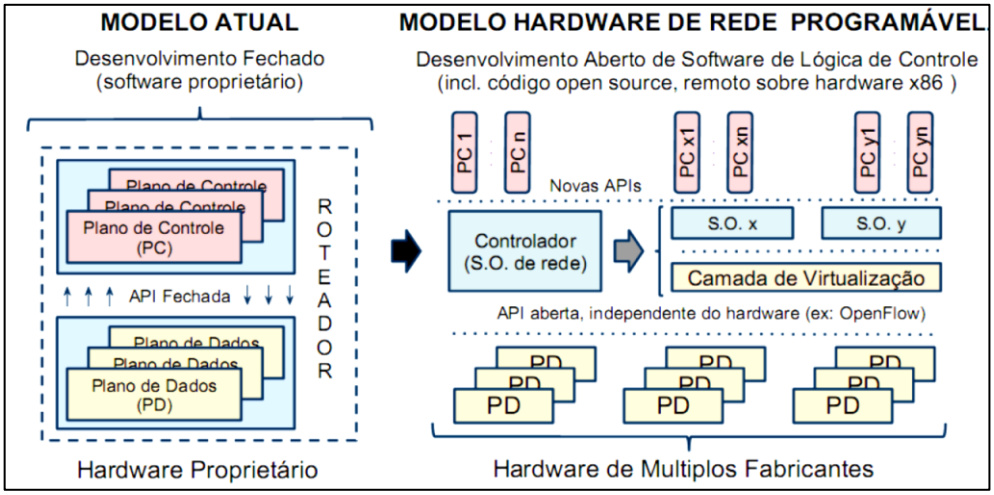
As Redes Definidas por *Software* constituem esse novo paradigma para o desenvolvimento das redes de computadores. As redes SDN abrem novas perspectivas em ambientes de controle lógico da rede, em novas aplicações de rede podendo ser desenvolvidas de forma simples e livre dos limites da arquitetura atual.

5.1 ARQUITETURA

O Modelo Atual (Figura 10) dos roteadores de rede é formado basicamente por duas camadas distintas: o *software* de controle e o *hardware* dedicado ao encaminhamento de pacotes. O *software* de controle é encarregado de tomar as decisões de roteamento, definindo a tabela de comutação para os roteadores. Este, por sua vez, transfere essas decisões à tabela de comutação, por meio de uma API (*Application Programming Interface*) proprietária para o *hardware* de encaminhamento, que realiza a comutação dos pacotes ao nível do *hardware*. A única interação de gerência do usuário, no caso o administrador de rede, com o dispositivo, ocorre por meio de interfaces de configuração Web ou SNMP, limitando-se ao uso de funcionalidades básicas programadas pelo fabricante.

Sendo o modelo atual definido por duas camadas autocontidas, não é necessário que elas sejam fechadas em um mesmo equipamento. A arquitetura das redes SDN, ou seja, o Modelo de *Hardware* de Rede Programável (Figura 10) subdivide essas camadas de forma que seja possível programar remotamente o dispositivo de rede, permitindo que a camada de controle possa ser movida para um servidor dedicado e com alta capacidade de processamento.

Figura 10 – Arquiteturas de roteadores: modelo atual e modelo programável



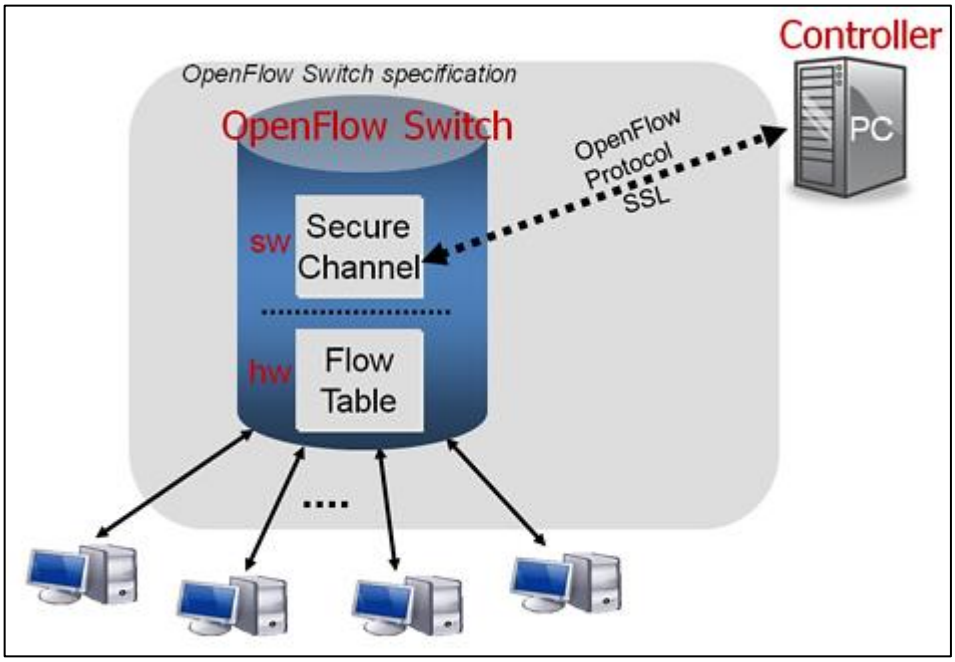
Fonte: Rothenberg et al. (2010).

Nesse alinhamento, mantém-se o alto desempenho no encaminhamento dos pacotes em *hardware* aliado à flexibilidade de inserir, remover fluxo de dados por meio de aplicações em *software* através de um protocolo aberto, API, para programação da lógica do equipamento.

5.2 ELEMENTOS PROGRAMÁVEIS DAS REDES SDN

De maneira mais específica, os elementos de comutação, *switches* e roteadores, exportam uma interface de programação que permite *softwares* de rede inspecionar, definir e alterar a tabela de roteamento do comutador. Isso acontece, por exemplo, nos comutadores OpenFlow, mostrado na Figura 11.

Figura 11 – Arquitetura do comutador OpenFlow



Fonte: Costa (2013).

O *software* em questão será, na prática, organizado com base em controladores de aplicação geral, nos sistemas operacionais de rede, que controlam aplicações específicas para a finalidade de cada rede. Outra possibilidade, e talvez uma das mais interessantes desse paradigma, é a capacidade de utilizar o FlowVisor, que permite que as aplicações de rede sejam divididas entre diferentes controladores.

O princípio básico das Redes Definidas por *Software* é a capacidade de programar os elementos de uma rede de computadores. Essa programação é a simples manipulação dos pacotes em um fluxo. Esse fluxo geralmente é definido em função dos recursos oferecidos pela interface de programação.

A maneira como os elementos de rede realizam a operação de encaminhamento dos pacotes é simples. Cada pacote, recebido em uma das interfaces do comutador de rede, é inspecionado e dele é gerado uma consulta a uma tabela de encaminhamento do comutador.

Atualmente, nos *switches* Ethernet, essa consulta é baseada no endereço MAC (*Media Access Control*) de destino do pacote, em roteadores IP, em um prefixo do endereço IP de destino.

Caso o comutador não encontre esse endereço na tabela de comutação, o pacote é descartado ou segue um comportamento padrão, como por exemplo, enviar esse endereço a todas as portas de saídas do comutador (conhecido com o termo de *broadcast*). Uma vez identificado, o destino do pacote, seja ele encontrado na consulta da tabela de comutação ou definido por um comportamento padrão, o mesmo atravessa as interconexões do comutador para atingir a porta de destino, onde ela entra em uma fila para a transmissão.

Ao passar do tempo, com a evolução das redes de computadores, o processo de consulta e chaveamento foi amplamente estudado no meio acadêmico, resultando em soluções baseadas usualmente em *hardware* com desempenho suficiente para acompanhar as taxas de transmissão do meio (GUEDES, 2012).

Nesse sentindo, as redes SDN têm a capacidade de controlar o plano de encaminhamento de pacotes por meio de uma interface bem definida. Sem dúvida, uma das interfaces mais conhecidas deste paradigma, desde o início, é o OpenFlow. O principal objetivo do OpenFlow é permitir que se utilize equipamentos de rede comercial para pesquisas de novos protocolos de rede em paralelo ao tráfego real de produção da rede.

Isso ocorre com os elementos de divisão de recursos, nos quais se define uma interface de programação que permite o desenvolvedor controlar diretamente a tabela de encaminhamento dos comutadores de pacotes presentes na rede. Esse tipo de proposta dá mais credibilidade à pesquisa para a indústria, já que as novas ideias serão validadas com o uso de comutadores de rede reais, de alto processamento e utilizadas em redes reais que, em geral, contêm grande fluxo de dados.

Apesar do OpenFlow ser o foco principal do paradigma SDN, não se limita apenas nele e a forma como ele expõe os recursos dos comutadores de rede, nem o exige como elemento essencial. Há diversas outras possibilidades de implementação de uma interface de programação que atenda os objetivos do paradigma.

Outra possibilidade de implementação é o conceito de interface de rede sNICh (RAM, 2010). O sNICh é uma implementação para ambientes em redes virtualizadas em que a divisão do plano de dados é feita de forma em que se possa dividir as tarefas de encaminhamento entre *host* e interface de rede. Com essa divisão é garantido uma eficiência no encaminhamento entre as máquinas virtuais no mesmo hospedeiro e a rede.

Nessa perspectiva, é possível imaginar uma interface definida para essa arquitetura que possa ser usada para o controle de roteamento por um software implementado em um

controlador de rede, que se apresente como uma opção para o uso de comutadores de software como os *switches* OpenFlow.

Ainda é possível implementar outras propostas para o paradigma SDN que alteram a divisão de tarefas entre o controlador e os *switches*. Isso é apresentado na proposta de arquitetura do DevoFlow (MOGUL, 2010). O DevoFlow aborda o argumento de que o OpenFlow é muito dependente do controlador de rede SDN.

No OpenFlow existe a necessidade de que todos os fluxos sejam acessíveis para o controlador que, por sua vez, impõe demandas sobre o *hardware* e limitações de desempenho que podem não ser aceitáveis em casos particulares e que podem ser definidos por regras simples.

Assim, o DevoFlow reduz o número de casos em que o controlador precisa ser acionado, aumentando a eficiência. Outras propostas de solução podem ser aplicadas ao OpenFlow, basta que se façam regras específicas para cada fluxo que sejam identificadas pelo controlador.

5.3 CONTROLADOR SDN

Sendo definida uma interface de programação dos comutadores de rede é necessário desenvolver uma aplicação que utilize essa interface para controlar cada *switch* separadamente. Esse desenvolvimento traz consigo limitações associadas ao *hardware* de cada equipamento. Esse desenvolvimento exige que o programador lide com tarefas de baixo nível no desenvolvimento de um *software* diretamente ligado a um elemento de *hardware* ocasionando erros e aumentando a complexidade. Além disso, novos desenvolvimentos exigem que todas as funcionalidades de baixo nível sejam reimplementadas.

Desse modo, faz-se necessário um novo nível na arquitetura das redes SDN, um nível que concentre as tarefas de manipulação dos elementos de rede oferecendo uma abstração de alto nível para o desenvolvedor, fazendo uma analogia clara aos sistemas operacionais de computadores pessoais. Essa é a definição da natureza do paradigma de SDN.

Esse componente, chamado de “sistema operacional de rede”, ou melhor, chamado de controlador SDN, pode concentrar a comunicação com todos os elementos programáveis da rede oferecendo uma visão unificada da rede. Por meio dele é possível desenvolver programas que, além de ter uma visão centralizada da rede, com análises detalhadas, é possível também implementar novas funcionalidades para chegar a decisões operacionais de como o sistema deve operar (CASADO et al., 2010), obtendo uma melhor gerência da rede.

Essa visão unificada da rede não necessariamente precisa ser centralizada. A analogia dos sistemas operacionais distribuídos pode ser implementada, do mesmo modo, nos controladores. A implementação pode ser desenvolvida de forma distribuída, seja pela divisão dos elementos ou por um controlador realmente desenvolvido, de forma distribuída com algoritmos que sejam capazes de manter uma visão consistente entre suas partes.

Foram desenvolvidos diversos controladores para o paradigma SDN. Muitos dos quais apresentam ambientes de tempo real de execução e que oferecem uma interface imperativa para programação da rede. O que determina, em grande parte, o estilo de desenvolvimento e as funcionalidades que o controlador oferece é a linguagem de programação em que ele foi desenvolvido, seja ele em C, Java, Python, Ruby, entre outros.

Vários controladores, já desenvolvidos, não se preocupam com requisitos de escalabilidade e disponibilidade da rede, optando, assim, por estruturas centralizadas. No entanto, há desenvolvimentos voltados para a implementação de amplos sistemas de redes nos quais, utilizam-se, diferentes formas de distribuição de controle para garantir requisitos como escalabilidade e disponibilidade do sistema. A Tabela 2 apresenta uma tabela comparativa dos principais controladores SDN.

Tabela 2 – Principais controladores SDN

Nome	**Linguagem**	**Código Aberto**	**Desenvolvedor**	**Característica**
NOX	C++/ Python	Sim	Nicira	É o primeiro controlador SDN, desenvolvido pela empresa Nicira e disponibilizado para a comunidade.
POX	Python	Sim	Nicira	Tem uma API SDN (interface de programação de aplicações) de alto nível, incluindo um gráfico de topologia e suporte para virtualização.
Maestro	Java	Sim	Rice University	É um controlador OpenFlow para orquestrar aplicações de controle de rede.
Beacon	Java	Sim	Stanford	É um controlador que suporta operação baseada em eventos.
Floodlight	Java	Sim	BigSwitch	É um controlador desenvolvido pela empresa Big Switch e disponibilizado para a comunidade como software livre.

Continua ...

Nome	Linguagem	Código Aberto	Desenvolvedor	Característica
Ryu	Python	Sim	Grupo NTT, OSRG	Um controlador que visa proporcionar controle logicamente centralizada e APIs para criar novos aplicativos de gerenciamento e controle da rede.
SNAC	C++	Não	Nicira	É atualmente o controlador da Nicira, que só disponibiliza o código binário, ele usa um gerenciador de política baseado na web para gerenciar a rede.
Trema	C/ Ruby	Sim	NEC	É um framework para o desenvolvimento de controladores OpenFlow.

Fonte: Costa (2013).

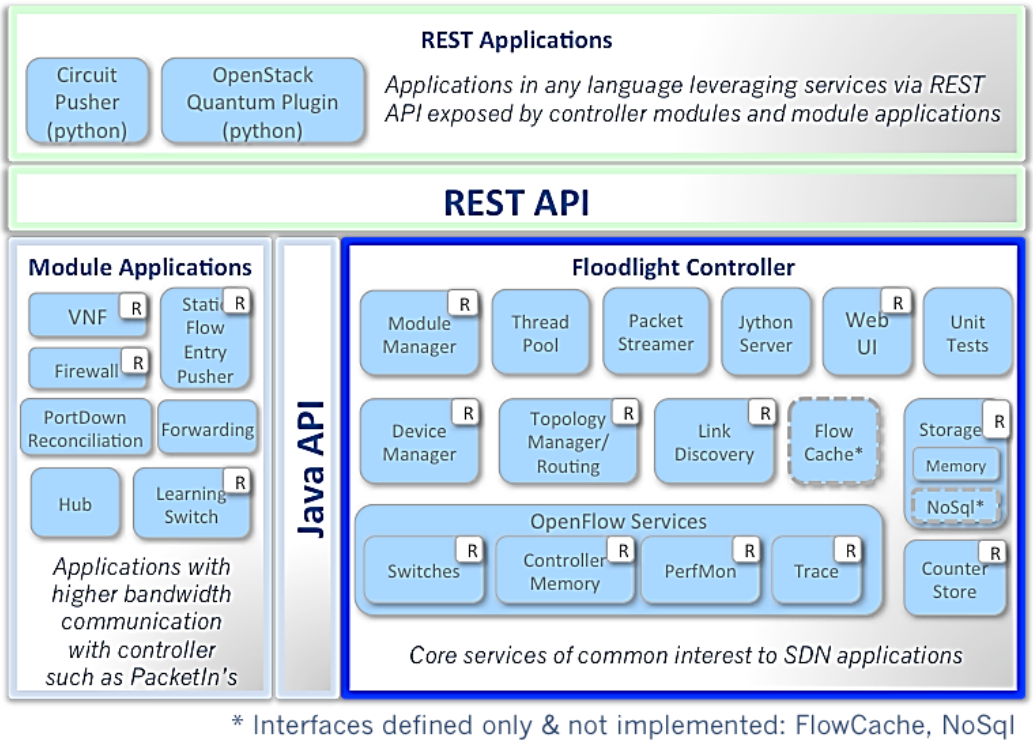
A escolha do controlador no desenvolvimento deste estudo foi feita baseando-se nas suas principais características. Para implementar o módulo, o controlador Floodlight foi selecionado por ser multiplataforma, código aberto e fornecer uma API totalmente baseada na linguagem Java, o que potencialmente aumentaria a produtividade da implementação. Além do Floodlight, os controladores Beacon e Maestro têm características similares e, portanto, poderiam ser utilizados. No entanto, preferiu-se utilizar o Floodlight por ser de classe empresarial, possuir uma boa documentação, ter uma ampla comunidade e fornecer uma interface amigável.

5.3.1 Floodlight

Floodlight é um controlador OpenFlow para redes corporativas que se originou do controlador Beacon e atualmente é apoiado pela *Open Networking Foundation* (ONF) e *Big Switch Networks*, que produz controladores comerciais que suportam o Floodlight (FLOODLIGHT, 2013). Em sua arquitetura, todos os elementos são módulos que exportam serviços. O controlador Floodlight suporta *switches* físicos e virtuais, podendo ainda manipular redes híbridas (OpenFlow e não OpenFlow), simultaneamente. O Floodlight oferece um sistema modular, o que facilita para o desenvolvedor estender e melhorar o código e é compatível com a ferramenta de simulação Mininet (MININET, 2016)

Além de ser um controlador OpenFlow, o Floodlight oferece uma coleção de aplicações desenvolvidas acima do controlador. Estas aplicações fornecem um conjunto de funcionalidades para atender a diferentes necessidades em uma rede OpenFlow, enquanto as funcionalidades internas do controlador compreendem um conjunto de serviços e módulos de interesse comum a qualquer aplicação SDN. A Figura 12 ilustra a relação entre o controlador e as aplicações.

Figura 12 – Controlador Floodlight: módulos de aplicações e aplicações REST



Fonte: Floodlight (2013).

O **controlador Floodlight** apresenta módulos que oferecem uma API que pode ser usada pelas Aplicações REST[9] (*Representational State Transfer*), destacado com um "R" dentro de um quadrado branco, no canto superior direito da representação gráfica da Figura 12. São eles:

a) **Module Manager:** responsável pelo carregamento de outros módulos, descritos em um arquivo de propriedades que é lido pelo controlador durante a inicialização. O Floodlight pode ser configurado para carregar diferentes módulos para acomodar aplicações variadas;

[9] REST é um estilo arquitetural que serve de alicerce para o desenvolvimento de aplicações em rede. A aplicação mais comum de REST é a própria *World Wide Web*.

b) **Thread Pool:** serve para fazer com que as tarefas sejam executadas em tempos específicos ou periodicamente;
c) **Packet Streamer:** é um serviço de fluxo de pacotes trocados entre qualquer *switch* e o controlador para um observador;
d) **Jython Server:** usado para depurar os programas escritos para o controlador, usando um interpretador *Jython* (implementação de *Python* em Java);
e) **Web UI:** interface gráfica via Internet para gerenciamento do controlador;
f) **Unit Tests:** testes de unidade que facilitam a busca por problemas no código de um módulo;
g) **Device Manager:** registra dispositivos conforme eles se movem pela rede, isto é, adições e remoções dos dispositivos na rede e define o dispositivo de destino para um novo fluxo;
h) **Topology Manager/Routing:** mantém a informação da topologia da rede para o controlador e encontra rotas na rede. São criadas "ilhas" OpenFlow que reúnem *switches* OpenFlow fortemente conectados numa mesma instância do Floodlight;
i) **Link Discovery:** é responsável em descobrir e manter o estado das ligações de dispositivos na rede OpenFlow. Este serviço usa o Protocolo de Descoberta da Camada de Enlace, conhecido como LLDP (*Link Layer Discovery Protocol*) e faz transmissão de pacotes para detectar as ligações;
j) **Flow Cache:** é uma API definida para o manejo de um conjunto de diferentes tipos de eventos na rede. Aplicações SDN frequentemente necessitam decidir quando e como tais eventos de rede devem ser tratados. Por exemplo, como manusear fluxos quando há uma falha em alguma ligação ou *switch*;
k) **Storage:** permite que módulos do Floodlight que implementam a interface *IStorage Source Service* possam criar, apagar e modificar dados nessa fonte de armazenamento, que fica em memória, no estilo NoSQL. Todos os dados armazenados nessa área são compartilhados, isto é, não há um mecanismo de proteção de memória que impeça um módulo de sobrescrever dados de outros módulos. E por permanecerem apenas em memória, caso a máquina seja desligada, dados nessa área são perdidos;
l) **Counter Store:** implementa um armazém central para contadores do sistema, como número de pacotes de entrada e de pacotes de saída, por exemplo. Esses contadores centrais não são afetados por outros módulos;

m) **OpenFlow Services (Switches, Controller Memory, PerfMon, Trace):** são os serviços que são oferecidos pelo protocolo OpenFlow e que são usados para se comunicar com os dispositivos compatíveis com o protocolo.

Do lado esquerdo da Figura 12, encontram-se os **módulos de aplicações,** que possuem maior vazão no canal de comunicação com o controlador, interagindo com ele por meio de uma API Java, disponibilizada por meio de numerosas classes de interface. Alguns módulos podem oferecer uma interface que pode ser usada por Aplicações REST. A Figura 12 apresenta as seguintes aplicações como:

a) **VNF (Virtual Network Filter):** uma simples aplicação isolada de rede baseada em MAC;
b) **Firewall:** uma aplicação para aplicar regras de permitir/negar o tráfego baseado nas especificações do experimento;
c) **Static Flow Entry Pusher:** módulo que instala e remove regras de fluxo em um *switch* específico;
d) **PortDown Reconciliation:** uma aplicação para reconciliar os fluxos por meio de uma rede quando uma porta ou *link* desliga;
e) **Forwarding:** módulo de encaminhamento de pacotes;
f) **Hub:** um aplicativo hub que inunda os pacotes de entrada para todas as portas ativas;
g) **Learning Switch:** um *switch* de aprendizagem na camada 2.

Finalmente, na parte superior da Figura 12, tem-se o conjunto de aplicações do controlador Floodlight que são as **aplicações REST** (aplicações que trocam mensagens HTTP) **Circuit Pusher** e **OpenStack Quantum Plugin**. Essas podem ser desenvolvidas em qualquer linguagem e se comunicam com o controlador Floodlight e com os módulos de aplicações por meio de uma API REST. O **Circuit Pusher** utiliza as APIs REST para criar um circuito bidirecional, isto é, uma entrada de fluxo permanente em todos os *switches* na rota entre dois dispositivos baseados em endereços IP com uma prioridade específica. É implementado com um script em Python. Já o **OpenStack Quantum Plugin** é um módulo de virtualização de rede da camada 2 (baseada em MAC). Assim, é possível criar múltiplas redes lógicas na camada 2 (camada de enlace de dados) com apenas um domínio. Esse módulo pode ser usado para implantar o OpenStack, que é uma plataforma de *software* de computação em nuvem.

O controlador Floodlight é executado por padrão na porta 6633 e provê uma interface com a Internet por meio da porta 8080, onde é possível consultar informações sobre os *switches*, *hosts*, fluxos e visualizar a topologia da rede SDN.

5.4 APLICAÇÕES

Seguindo o presente contexto, considerando os controladores do paradigma SDN como sistemas operacionais de rede, o *software* desenvolvido para criar novas funcionalidades pode ser visto como uma aplicação que é executada sobre uma rede física.

Dessa forma, observando a rede como um sistema unificado e gerenciado por um sistema operacional, pode-se, por exemplo, implementar soluções de roteamento e de encaminhamento especialmente desenhadas para ambientes particulares, controlando as interações entre os diversos comutadores da rede.

A flexibilidade que esse paradigma oferece para estruturar sistemas de rede é útil em praticamente todas as áreas de aplicação das redes de computadores. Sua estrutura lógica centralizada permite o desenvolvimento de novas funcionalidades de maneira fácil e bem diversificado, abrangendo assim uma ampla diversidade de ambientes de rede. Considerando que as redes SDN definem essa nova forma de estrutura, pode-se avaliar que ela seja aplicável a qualquer tipo de ambiente no cenário de redes de computadores, beneficiando uma melhor organização das funcionalidades oferecidas em torno de uma visão lógica completa da rede, em especial, para o gerenciamento de energia.

O gerenciamento de energia, do mesmo modo, é um dos pontos fortes que se obtém de maneira fácil com o paradigma SDN. Atualmente, as soluções que conservam e gerenciam energia em sistemas da computação vêm crescendo amplamente. Em redes de grande porte, nas quais os recursos destas consomem uma parcela significativa do gasto de energia, reduzir o consumo por meio de comunicação, torna-se uma possibilidade interessante.

Em determinados períodos do dia, alguns elementos da rede ficam ociosos em virtude do baixo fluxo de dados na rede. Os desligamentos desses dispositivos de rede tornam-se viáveis devido à baixa taxa de transmissão dos mesmos provocando a redução do consumo de energia nessa rede.

Isso é possível devido à característica da visão global da rede que é provida pelo paradigma SDN. Uma vez dispondo dessa visão global, torna-se fácil a identificação desses elementos ociosos e até mesmo a redefinição de rotas, a fim de desviar o tráfego de elementos passíveis de desligamento. Além disso, o uso do controle de rotas de decisão permite a implementação de pontos de controle que podem interceptar tráfego “ruidoso” na rede, evitando que pacotes de menor importância atinjam os elementos subutilizados, passíveis de desligamento, evitando que os mesmos sejam ativados desnecessariamente.

6 DESENVOLVIMENTO DO MÓDULO ECONOMIA DE ENERGIA NO CONTROLADOR FLOODLIGHT PARA EFICIÊNCIA ENERGÉTICA NOS SWITCHES EM REDES OPENFLOW

Proprietários de *data centers* sugerem o uso do conceito de Redes Definidas por *Software* e OpenFlow como uma nova forma de reduzir a complexidade da rede do *data center* e consumo de energia. A especificação OpenFlow usa um controlador externo (por exemplo, Floodlight) que trata as decisões de roteamento. Essa abordagem oferece flexibilidade para o projeto de rede do *data center* a fim de fornecer uma confiável, escalável e econômica infraestrutura computacional para serviços massivos da Internet. Com base em um alto nível de programação e isolamento, OpenFlow reduz o consumo de energia e os custos operacionais, melhorando a eficiência energética.

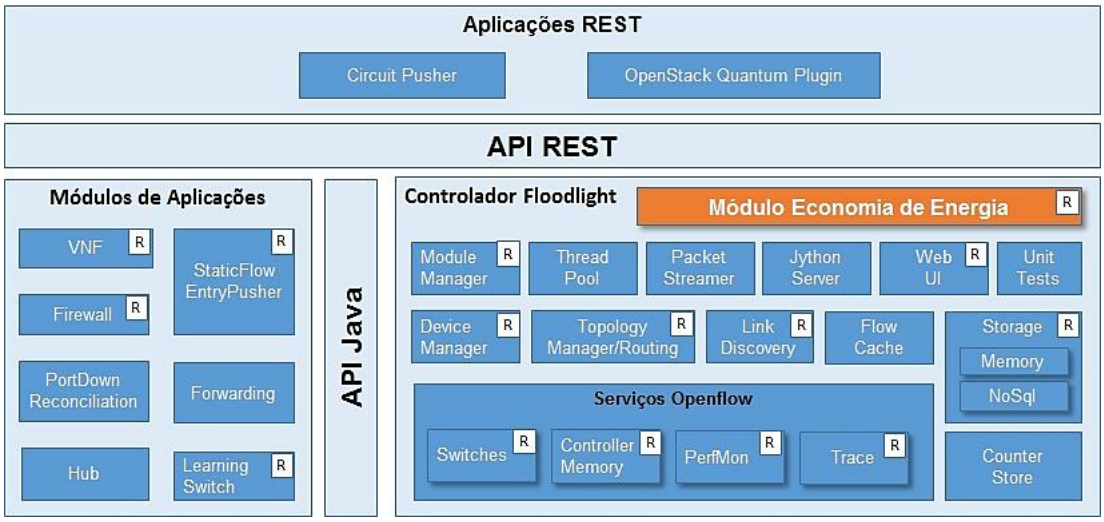
Na subseção seguinte será apresentado o Módulo Economia de Energia, suas principais características e o fluxo de execução para o gerenciamento de energia nos *switches* em redes OpenFlow.

6.1 O MÓDULO ECONOMIA DE ENERGIA

Neste trabalho, o Floodlight (PROJECT FLOODLIGHT, 2015) foi usado para implementar o módulo, devido às seguintes características: possuir código aberto, ser escrito em Java, suportar múltiplos fluxos de execução, incluir módulos predefinidos para representar o estado dos elementos em uma rede OpenFlow e implementar funcionalidades básicas de rede.

O módulo desenvolvido denominado Economia de Energia foi inserido no controlador Floodlight e oferece uma API que pode ser usada pelas Aplicações REST, apresentado na Figura 13. No Apêndice A é exibido o algoritmo que mostra as etapas de elaboração deste módulo.

Figura 13 – Controlador Floodlight com o Módulo Economia de Energia



Fonte: Adaptado de Floodlight (2013).

O Módulo Economia de Energia é composto de arquivos de classes Java, propriedades e REST, além de possuir interação com outros módulos do controlador Floodlight. Neste módulo está contido:

a) Carregamento de arquivos com valores de custos[10]para as ligações dos *switches* de redes OpenFlow. No módulo foram criados dois arquivos: "LinkCustoInicial.properties" (Apêndice B) invocado pelo Módulo Economia de Energia na inicialização do controlador, apresentando a configuração inicial dos custos e o script "LinkNovoCusto.sh" (Apêndice C) que mostra uma nova composição de custos quando for requisitado pelo próprio controlador ou administrador de rede;

b) Criação da Árvore de Extensão Mínima, por meio do algoritmo MiNet sobre as ligações dos *switches*, exibido no Apêndice D. MiNet é utilizado para calcular a árvore no grafo atualizado. Em detalhe, as implementações da topologia são realizadas por listas de adjacência e/ou matrizes de adjacência; o código utiliza filas de prioridade, estruturas de dados e custos nas ligações para encontrar a união em reduzir o tempo de complexidade para O(ElogV), cujo E é o número de ligações entre os *switches* e V é o número de *switches*. Escolhe-se um *switch* (randomicamente ou pelo administrador de rede), o algoritmo observa o conjunto de ligações possíveis com aquele dispositivo selecionado e elege as ligações que não formam ciclos com o subgrafo, cujo custo é o mínimo possível naquele momento.

[10] O valor de custo será correspondente à velocidade da porta do *switch*.

Se uma ligação apresentar todos estes quesitos, pode-se considerá-la segura. Neste caso, o algoritmo MiNet, fornecerá uma resposta ótima para o problema de eficiência energética nos *switches* que não necessariamente é única. As seleções de ligações serão realizadas até que todos os dispositivos estejam conectados. Logo, serão enviadas mensagens de comandos OpenFlow do Módulo Economia de Energia para as portas Ethernet dos *switches* que ficaram fora da árvore. Mensagens de comandos com o modo da porta (OFP_PORT_MOD) desativado (OFPPC_PORT_DOWN);

c) Exibição de RESTs (Apêndice E) que coletam informações para:
   - Listar todos os endereços MAC dos *switches* conectados ao controlador (por exemplo, 00:00:00:00:00:00:00:01, correspondente ao endereço MAC do *switch* s1),
   - Mostrar ligações entre os *switches* e seus custos (por exemplo, {s1-s3}=10, ou seja, *switch* s1 com ligação em s3, possui custo 10),
   - Exibir detalhadamente as ligações entre os *switches*, portas Ethernet e os custos aplicados (por exemplo, {s1:eth1-s3:eth1}=10, ou seja, *switch* s1 na porta Ethernet eth1 com ligação em *switch* s3 na porta Ethernet eth1, possui custo 10),
   - Apresentar somente as ligações entre os *switches*, portas Ethernet e custos, livres de ciclos identificados pelo MiNet,
   - Exibir as ligações entre os *switches*, portas Ethernet e custos, que formam ciclos apontado pelo MiNet.

O arquivo de configuração “floodlight.properties” que especifica quais módulos devem ser carregados na inicialização do controlador foi alterado para inclusão do Módulo Economia de Energia. No arquivo, é incluído um valor de chave em uma única linha. O Apêndice F mostra essa inserção. Para encontrar os módulos no *classpath*[11] usa-se o carregamento de serviço (ServiceLoader) do Java. É requerido para listar todas as classes que implementam “IFloodlightModule” no arquivo “net.floodlightcontroller.module.IFloodlightModule”. Nesse caso, insere o nome completo do módulo em uma própria linha. O Apêndice G lista o arquivo alterado. O arquivo “logback.xml”, apresentado no Apêndice H, do mesmo modo, foi alterado para mostrar as mensagens de depuração do Módulo Economia de Energia.

Outros arquivos importantes foram importados do pacote “java.util” no arquivo de classe do módulo.

[11] Entende-se por *classpath* um parâmetro que indica à Máquina Virtual Java (JVM) onde procurar pacotes e classes definidos pelo usuário.

a) **ArrayList:** adiciona na topologia custos nas ligações entre os *switches*;
b) **Collection**: onde estão adicionados todos os serviços do módulo;
c) **HashMap:** é uma tabela de mapeamento (pares, chave e valor) que determina onde armazenar e procurar os dados dos *switches*;
d) **HashSet:** é uma tabela de conjuntos de componentes que decide onde registrar e buscar os dados dos *switches*;
e) **List:** lista as mudanças na topologia pelo objeto "linkUpdates";
f) **Vector:** implementa uma matriz com os custos nas ligações entre os *switches*.

A seguir, serão listados os principais módulos e interfaces do controlador Floodlight importados, da mesma forma, no arquivo de classe do Módulo Economia de Energia:

a) **IFloodlightProviderService:** é uma interface de núcleo do controlador que permite a interação com *switches* conectados;
b) **IOFSwitch:** é uma interface de núcleo do controlador que oferece métodos para interagir com *switches* usando comandos OpenFlow e, recuperando informações sobre os mesmos;
c) **FloodlightModuleContext:** é um módulo de núcleo do controlador que oferece serviço de registro para uma interface "IFloodlightProvider";
d) **IFloodlightModule:** carrega os módulos nativos e os desenvolvidos por pesquisadores no Floodlight;
e) **IFloodlightService:** é uma interface base de núcleo do controlador para qualquer pacote "IFloodlightModule" que fornece um serviço;
f) **ILinkDiscovery (Link Discovery):** responsável por descobrir e manter o estado das ligações na rede OpenFlow;
g) **LDUpdate (Link Discovery):** "objetolinkUpdates" que envia mensagem de notificação de mudança da topologia;
h) **ITopologyService (Topology Manager):** mantém a informação da topologia para o controlador, assim como, encontrar caminhos na rede;
i) **ITopologyListener (Topology Manager):** todas as informações sobre a topologia atual são armazenadas em uma estrutura de dados chamado de instância de topologia. Se existir alguma alteração, uma nova instância é criada e a mensagem de notificação de mudança na topologia é invocada;
j) **IRestApiService (Rest Server):** executa um servidor de API REST.

6.2 CARACTERÍSTICAS DO MÓDULO ECONOMIA DE ENERGIA

O Módulo Economia de Energia tem como principais características as ações de:

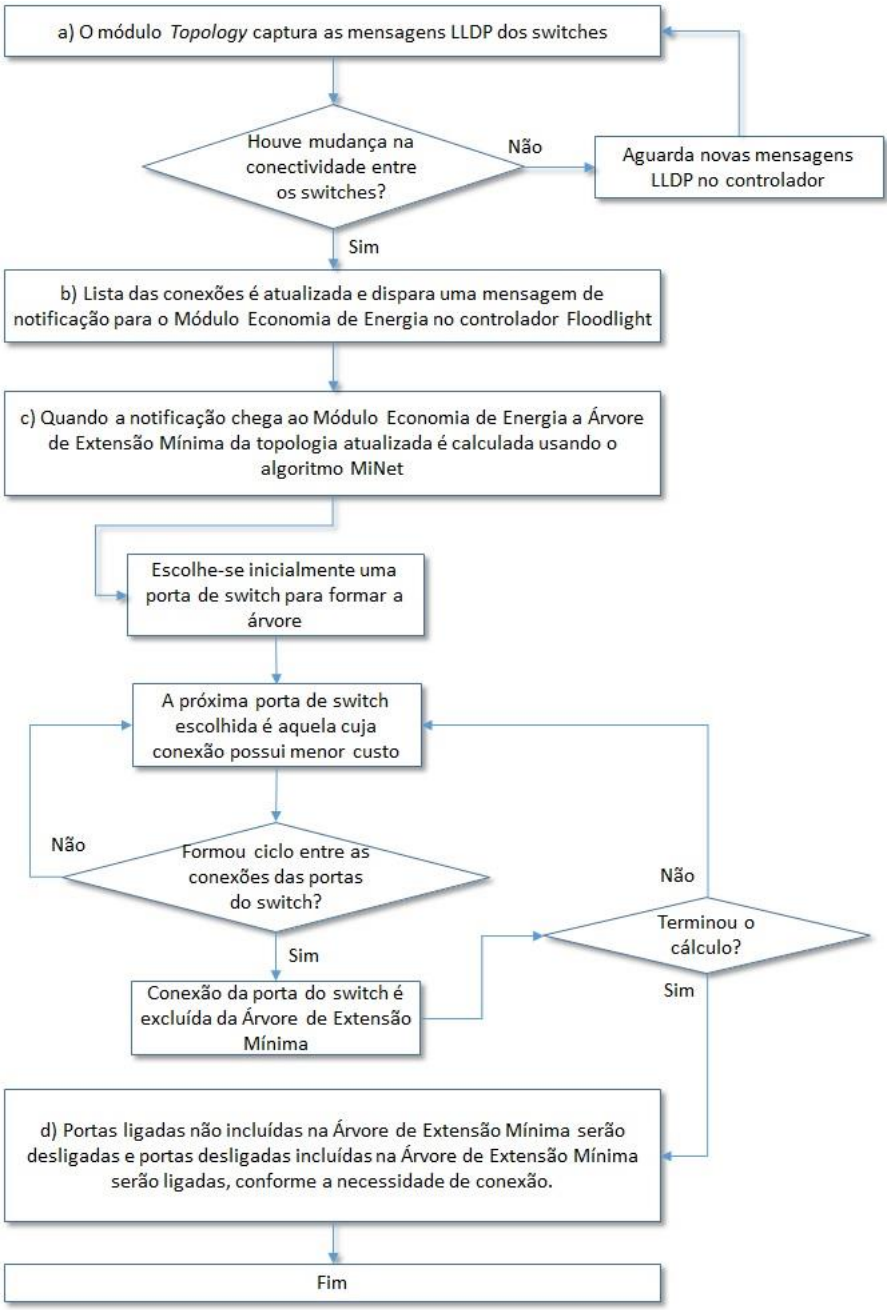
a) Receber notificações de mudanças na topologia proposta (ligações adicionadas ou removidas, custos sobre as ligações nos *switches*, entre outros);
b) Calcular uma Árvore de Extensão Mínima (algoritmo MiNet) para desativar as portas Ethernet dos dispositivos que estão fora da árvore, por intermédio de comandos OpenFlow, com o objetivo de reduzir o consumo de energia nos *switches*. O módulo, do mesmo modo, ativa as portas Ethernet, quando necessário, se ocorrer fluxo de dados excedente ou novos custos nas ligações dos *switches*;
c) Em seguida, os *hosts* podem enviar/receber o tráfego de dados, no decurso das ligações ativas.

Nesta perspectiva é que se sustenta o módulo fundamental de inovação na presente tese.

6.3 FLUXO DE EXECUÇÃO DO MÓDULO ECONOMIA DE ENERGIA

O fluxograma da Figura 14, exibe as ações desencadeadas a partir do momento em que o controlador recebe notificações da topologia proposta:

Figura 14 – Fluxograma de ações após mudanças na conectividade entre os *switches*



Fonte: Elaboração da própria autora.

As ações do fluxograma (Figura 14) referente ao Módulo Economia de Energia serão explicadas detalhadamente para compreender o fluxo de execução.

a) O controlador Floodlight (Figura 13) armazena o estado instantâneo da rede, utilizando o módulo *Topology Manager/Routing*. Pacotes LLDP são enviados do módulo *Link Discovery* para o módulo *Topology* e, este por sua vez, envia os pacotes LLDP para os *switches* que retornam informações de conectividade;

b) A observação do protocolo LLDP é usada para capturar as mudanças na conectividade entre os *switches*. Quando o módulo *Topology* registra mudança (em detalhe, o módulo *Topology* escuta o estado das portas e os pacotes de entrada. Um filtro é aplicado aos pacotes de entrada para detectar o tráfego LLDP), a lista das conexões é atualizada e uma mensagem de notificação (mudança de estado na ligação ponto-a-ponto entre as interfaces dos dispositivos) é enviada para os outros módulos do Floodlight. A mensagem de notificação igualmente chega ao Módulo Economia de Energia relatando os endereços e as interfaces dos *switches* modificados, por exemplo, *switch* s1 (endereço MAC 00:00:00:00:00:00:00:01, porta eth1, estado da porta: ativo) e *switch* s3 (endereço MAC 00:00:00:00:00:00:00:03, porta eth1, estado da porta: ativo) possuem custo 10 na ligação entre eles (s1:eth1-s3:eth1=10);

c) O Módulo Economia de Energia ativa a sua sequência por reagir as invocações das notificações e mensagens do estado das portas dos *switches*. Este módulo usa as informações do objeto *linkUpdate* (identificações de dispositivos e interfaces) para manter uma representação da topologia de rede atualizada, indicando quais ligações estão ativas. Por meio do algoritmo MiNet, deve-se construir uma Árvore de Extensão Mínima sobre os *switches* OpenFlow, que não formem ciclos entre os seus vértices;

d) Após o cálculo da Árvore de Extensão Mínima, um conjunto de mudanças é avaliado para minimizar o número de interações entre o controlador e os *switches*. O conjunto de mudanças mantém a lista de ligações que mudaram com a árvore. Interfaces são ativadas/desativadas de acordo com o conjunto de mudanças, por meio do envio de mensagens do Módulo Economia de Energia ao módulo *Topology*, que por sua vez, propaga para a rede as mensagens com os comandos OpenFlow para modificar o comportamento da (s) porta (s) física (s) para ativado/desativado.

Da mesma forma, o módulo *Device Manager* mantém a lista de como os *hosts* estão ligados aos *switches* OpenFlow, logo após ativar ou desativar as ligações. Cada vez que chega um pacote ao módulo, os cabeçalhos são analisados e os endereços da camada 2 são acoplados ao *switch* correspondente. O papel fundamental do controlador é agregar todas as tabelas em uma área de memória única (módulo *Storage*), simplificando a operação de pesquisa que associa a ligação de um *switch* a um endereço MAC de *host* de destino. O módulo do *switch* de aprendizagem (*Learning Switch*) fará o envio e a recepção dos pacotes pela rede, no trajeto das ligações ativas.

À medida que o controlador é acionado em cada mensagem LLDP, o Módulo Economia de Energia constrói uma Árvore de Extensão Mínima estável. Os custos sobre os arcos do grafo da topologia podem ser ajustados de acordo com as necessidades da aplicação. Por exemplo, informações sobre a largura de banda disponível ou a latência média na ligação. Para simplificar, durante os testes foram consideradas as ligações que têm o menor custo. Nesse caso, a fila de prioridade utilizada pelo algoritmo MiNet considera o número MAC dos *switches* como critério de ordenação a partir da escolha do vértice inicial.

Na seção7, serão explanados estudos de simulação em rede OpenFlow, empregando o uso do conceito de Redes Definidas por *Software* com o controlador Floodlight em um ambiente virtualizado com o ambiente de experimentação GENI. Os dados resultantes deste *testbed* com o Módulo Economia de Energia serão avaliados e comparados para o gerenciamento de energia nos *switches*. A topologia de rede será Fat Tree (árvore gorda), sendo uma das mais utilizadas em *data centers* por sua eficiência.

7 EXPERIMENTAÇÃO: EMULAÇÃO E TESTES NA FEDERAÇÃO GENI

Nesta seção será exibida a plataforma de testes, resultados e discussão comparando a eficiência energética do Módulo Economia de Energia no escalonamento da rede (dez e vinte *switches*) em três simulações para constatar as ações de economia de energia do módulo desenvolvido:

a) **Primeira simulação:** com o controlador Floodlight na configuração padrão, ou seja, sem o Módulo Economia de Energia;
b) **Segunda simulação:** com o controlador Floodlight e o Módulo Economia de Energia aplicando a configuração de custos iniciais nas ligações entre os *switches* OpenFlow;
c) **Terceira simulação:** com o controlador Floodlight e o Módulo Economia de Energia reconfigurando novos custos nas ligações entre os *switches* OpenFlow.

7.1 PLATAFORMA DE TESTES

No ambiente de experimentação GENI foi criado o projeto intitulado como "ProjetoUnesp" e duas fatias designadas como FatTree10 e FatTree20; o orientador deste trabalho é o líder do projeto e a autora é membro. Para os testes do Módulo Economia de Energia foi empregada a agregação da Universidade de UtahDDC (Utah Downtown Data Center) no *rack* InstaGENI. A Figura 15 apresenta a localização dos recursos alocados.

Figura 15 – Localização dos recursos alocados na Universidade UtahDDC

Fonte: Adaptado de GENI (2016).

A agregação InstaGENI inclui *hosts* (servidores com virtualização Xen e máquinas dedicadas) que podem carregar experimentos específicos de imagens de sistemas operacionais. O *rack* também suporta topologias OpenFlow.

O *rack* InstaGENI oferece um pequeno cluster ProtoGENI com rede OpenFlow e gerenciamento da agregação. Este *rack* inclui os seguintes componentes de sistemas, apresentados na Figura 16:

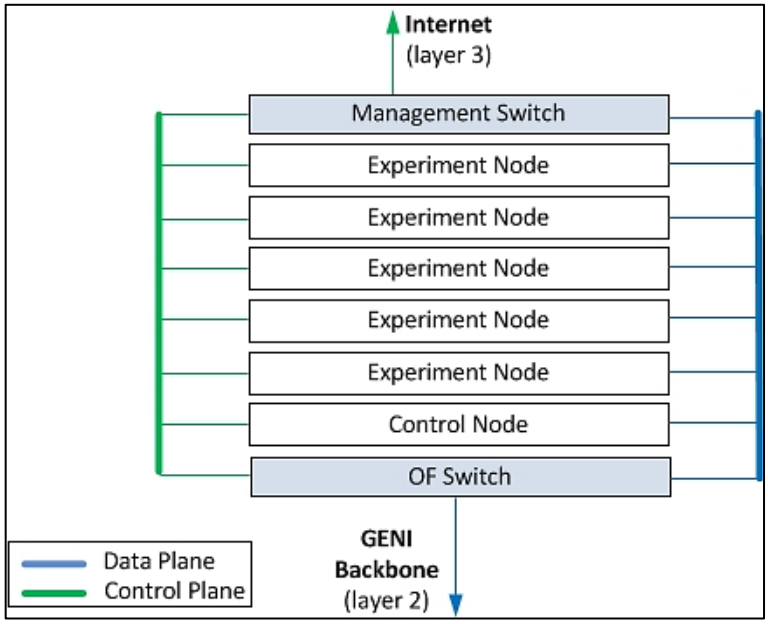
a) **Nó de controle:** servidor Xen que executa máquinas virtuais para prover:
   - Nó ProtoGENI, servidor web, monitoramento e servidor API de gerenciamento de agregação,
   - Nó servidor de arquivo local.

b) **Nós experimentais:** nós gerenciados pelo software ProtoGENI, que fornece serviços de inicialização, criação de contas, gestão experimental, etc.;

c) ***Switch* OpenFlow:** fornece roteamento interno e conectividade no plano de dados com o *backbone* de GENI (camadas 2 e 3);

d) ***Switch* gerenciável:** fornece conectividade do plano de controle com a Internet (camada 3).

Figura 16 – Componentes de rede para um *rack* InstaGENI



Fonte: GENI (2016).

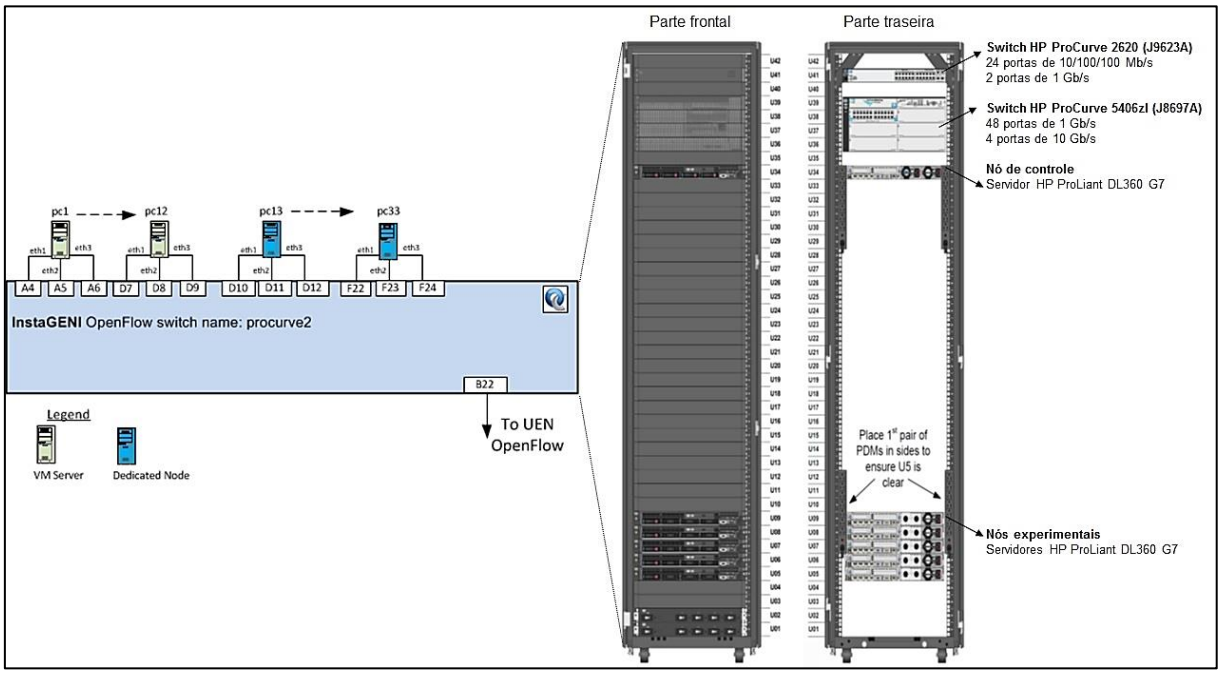
A atual especificação de *hardware* dos componentes para o *rack* InstaGENI, mostrados na Figura 17, inclui:

a) **Recurso de máquinas:** servidor da série HP que proporciona servidores de máquinas virtuais, monitoramento, armazenamento e funções das aplicações:
   - **Nó de controle**: servidor HP ProLiant DL360 G7 ou DL360e G8, quad-core, single-socket, 12 GB Ram, disco rígido de 4 TB (RAID) e duplo NIC,
   - **Nós experimentais:** servidores HP ProLiant DL360 G7 ou DL360e G8, six-core, dual-socket, 48GB Ram, disco rígido de 1 TB e duplo NIC.

b) **Componentes de rede:** fornece acesso as redes principais de GENI e Internet convencional:
   - **Internet convencional:** *Switch* HP ProCurve 2620 (J9623A) com 24 portas de 10/100/1000 Mb/s e 2 portas de 1 Gb/s,
   - ***Backbone* do GENI:** *Switch* HP ProCurve 5406zl (J8697A) com 48 portas de 1 Gb/s e 4 portas de 10 Gb/s.

O rack InstaGeni da Universidade de UtahDDC possui 33 nós, sendo, 12 servidores para máquinas virtuais e 21 PCs dedicados.

Figura 17 – *Rack* InstaGENI na Universidade de UtahDDC



Fonte: Adaptado de GENI (2016).

Por meio da API de gerenciamento de agregação foram solicitados os recursos no *rack* InstaGENI:

a) A topologia de rede escolhida é a Fat Tree. Ela consiste basicamente em duas ou três camadas de *switches*, composta por núcleo (topo), agregação (intermediária) e borda

(base). Para as duas fatias criadas no Projeto Unesp, as topologias são em três camadas;

b) Fatia FatTree10 (Figura 18):
   - Dez *switches* virtuais (pcvm – *personal computer virtual machine*) emulados pelo Xen[12]: s1 (pc10), s2 (pc7), s3 (pc3), s4 (pc11), s5 (pc6), s6 (pc2), s7 (pc7), s8 (pc7), s9 (pc7) e s10 (pc7), pcX é identificação do "pc" no *rack* InstaGENI. Todos equipados com o software Open vSwitch (versão 2.3.1), sistema operacional Ubuntu14-OVS2.3, apresentando seis portas Ethernet virtuais de conexão em cada um;
   - Oito *hosts* virtuais (pcvm) emulados pelo Xen: h1 (pc13), h2 (pc14), h3 (pc4), h4 (pc1), h5 (pc8), h6 (pc5), h7 (pc12) e h8 (pc15), pcX é identificação do pc no *rack* InstaGENI. Os *hosts* possuem sistema operacionalUbuntu14-64-STD e uma porta Ethernet virtual em cada *host* para se comunicar com os *switches* de borda da topologia.
   - Composição da rede FatTree:
     - Núcleo: s1 e s2;
     - Agregação: s3, s4, s7 e s8;
     - Borda: s5, s6, s9 e s10.
   - Largura de banda:
     - Ligações entre os *switches* e o controlador: 40 Mbps (megabits por segundo);
     - Ligações entre os *switches* de núcleo e agregação: 10 Mbps;
     - Ligações entre os *switches* de agregação e borda: 08 Mbps;
     - Ligações entre os *switches* de borda e os *hosts*: 02 Mbps.
   - Em relação ao controlador c0 (pc7) será o Floodlight na versão 1.0 com o Módulo Economia de Energia incluído e emulado pelo Xen.

c) Fatia FatTree20 (Figura 19):
   - Vinte switches virtuais (pcvm) emulados pelo Xen[13]: s1 (pc10), s2 (pc10), s3 (pc15), s4 (pc15), s5 (pc10), s6 (pc15), s7 (pc5), s8 (pc4), s9 (pc15), s10 (pc15), s11 (pc15), s12 (pc15), s13 (pc10), s14 (pc15), s15 (pc10), s16 (pc11), s17 (pc10), s18 (pc15), s19 (pc8) e s20 (pc10). Todos equipados com o software Open vSwitch (versão 2.3.1), sistema operacional Ubuntu14-OVS2.3, apresentando seis portas Ethernet virtuais de conexão em cada um;

[12] O hipervisor Xen hospeda as máquinas virtuais.
[13] O hipervisor Xen hospeda as máquinas virtuais.

- Dezesseis *hosts* virtuais (pcvm) emulados pelo Xen: h1 (pc5), h2 (pc14), h3 (pc4), h4 (pc12), h5 (pc7), h6 (pc3), h7 (pc1), h8 (pc6), h9 (pc10), h10 (pc10), h11 (pc11), h12 (pc11), h13 (pc8), h14 (pc13), h15 (pc2), h16 (pc10), pcX é identificação do pc no *rack* InstaGENI. Os *hosts* possuem sistema operacional Ubuntu14-64-STD e uma porta Ethernet virtual em cada *host* para se comunicar com os *switches* de borda da topologia;
- Composição da rede FatTree:
  - Núcleo: s1, s2, s3 e s4,
  - Agregação: s5, s6, s9, s10, s13, s14, s17 e s18,
  - Borda: s7, s8, s11, s12, s15, s16, s19 e s20.
- Largura de banda:
  - Ligações entre os *switches* e o controlador: 40 Mbps;
  - Ligações entre os *switches* de núcleo e agregação: 10 Mbps;
  - Ligações entre os *switches* de agregação e borda: 08 Mbps;
  - Ligações entre os *switches* de borda e os *hosts*: 02 Mbps.
- Em relação ao controlador c0 (pc15) será o Floodlight na versão 1.0 com o Módulo Economia de Energia incluído e emulado pelo Xen.

d) Para a produção (criação, recepção e análise) do tráfego de dados que atravessa toda a infraestrutura do sistema de comunicação da rede e será usado para analisar os resultados foi empregado o D-ITG (*Distributed Internet Traffic Generator*) versão 2.8.1. Ele gera tráfego de vários caminhos e é capaz de coletar estatísticas relevantes (Apêndice I);

e) O acesso aos recursos dos *switches*, *hosts* e controlador será realizado pelo programa SSH. O GENI fornece chaves, pública e privada, para serem instaladas na máquina que fará o acesso a federação.

Em geral, um *switch* é um elemento na rede com múltiplas portas que processa e encaminha os dados na camada de enlace de dados (camada 2). Foi adicionado uma ponte (conhecido também com o termo *Bridge*) em cada Open vSwitch e anexadas portas em cada ponte. Para controlar os fluxos entre os *hosts*, a criação da ponte é relevante para as interfaces conectadas a eles para se identificarem na rede. No Quadro 1 é mostrado, como exemplo, a criação de ponte no *switch* s5.

Quadro 1 – Criação de ponte no *switch* s5

```
root@s1:/users/ligiapre# ovs-vsctl add-br s5
ovs-vsctl add-port s5 eth1
ifconfig eth1 0
ovs-vsctl add-port s5 eth2
ifconfig eth2 0
ovs-vsctl add-port s5 eth3
ifconfig eth3 0
ovs-vsctl add-port s5 eth4
ifconfig eth4 0
ovs-vsctl add-port s5 eth5
ifconfig eth5 0
```

Fonte: Elaboração da própria autora.

Outra configuração importante no Open vSwitch é conhecer as informações do controlador. Com o comando do Quadro 2, a ponte estará se conectando a um controlador OpenFlow em 172.17.7.4, que está escutando na porta 6633:

Quadro 2 – Conexão da ponte do *switch* s5 com o controlador

```
root@s1:/users/ligiapre# ovs-vsctl set-controller s5 tcp:172.17.7.4:6633
```

Fonte: Elaboração da própria autora.

Os *switches* configurados em modo "*bridge*", ou seja, uma ponte Ethernet, interconectam as interfaces do "pc" (Figuras 18 e 19) às interfaces das máquinas virtuais. Nesse caso, as interfaces do "pc" são representadas por interfaces virtuais que são ligadas ponto-a-ponto com as interfaces dos "pc". O comutador por software (Open vSwitch) faz o encaminhamento dos pacotes de acordo com o endereço do "pc" no qual está a interface virtual. Quando um pacote chega, a primeira ação tomada é gravar, em uma tabela de aprendizagem, a porta do comutador e o endereço de origem do pacote, marcando que tal endereço é acessível pela porta de entrada do pacote. Para determinar a rota para o pacote, o Open vSwitch consulta o controlador OpenFlow, que agrega uma entrada na tabela de encaminhamento de fluxos do Open vSwitch e o pacote é encaminhado ao "pc". O processo é repetido até que o pacote chega ao "pc", que possui a máquina virtual e o pacote é entregue usando a interface virtual do Xen.

O Open vSwitch agrega a cada pacote uma etiqueta de VLAN para isolar as redes virtuais. Dessa forma, cada rede virtual possui uma etiqueta de VLAN diferente e única.

Em resumo, um pacote saindo de uma interface virtual passa pelo mecanismo de entrada/saída do Xen, o Open vSwitch agrega a etiqueta de VLAN é a rota é consultada com o controlador OpenFlow. Finalmente o pacote é encaminhado para o próximo salto. No “pc” de destino, o Open vSwitch retira a etiqueta VLAN e o Xen entrega o pacote para a máquina virtual correspondente.

Em relação aos *hosts* das duas topologias (Figuras 18 e 19), uma vez que estão em subredes diferentes eles precisam saber para onde enviar os pacotes adicionando uma entrada à tabela de roteamento (Quadro 3):

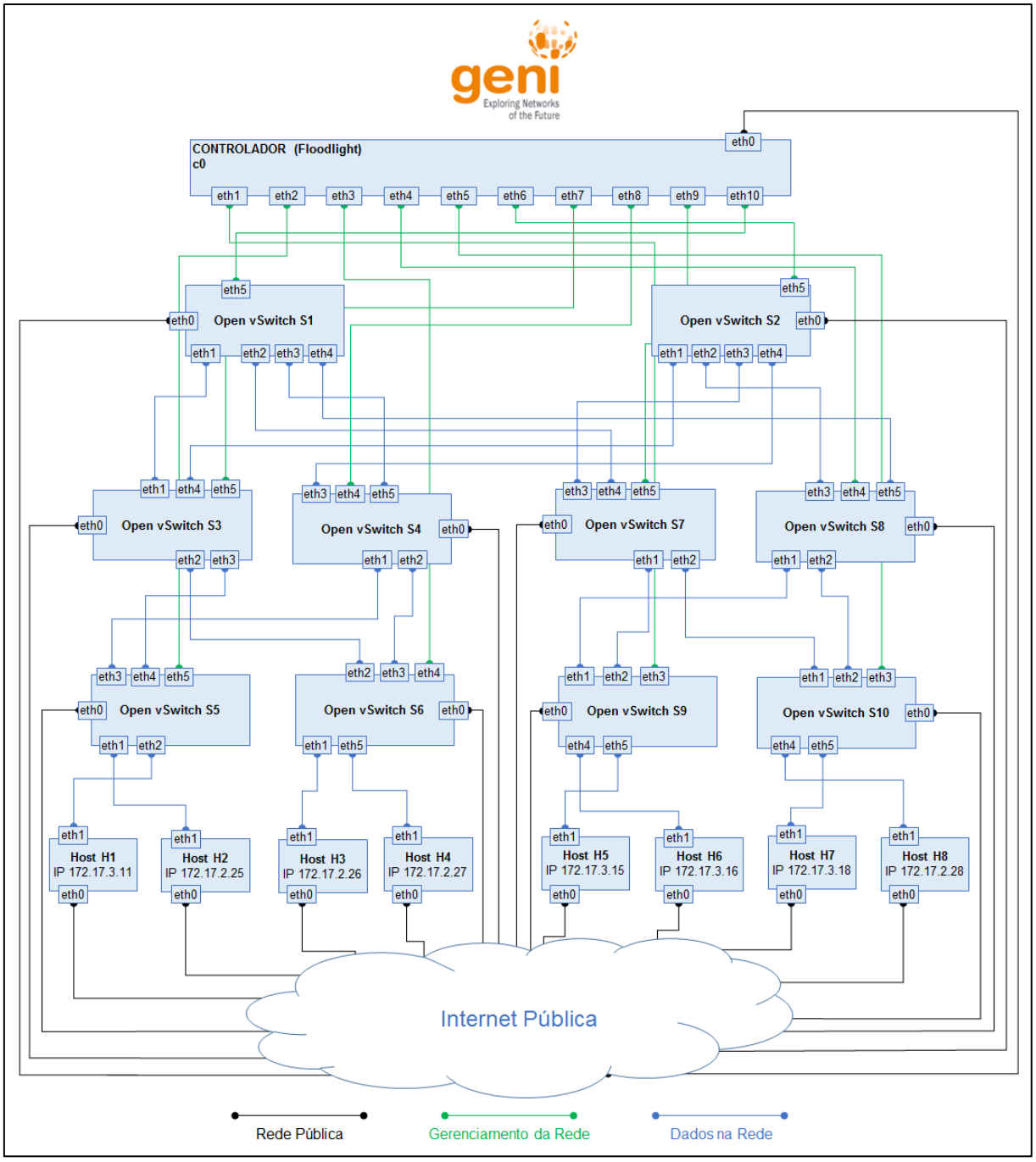
Quadro 3 – Entrada na tabela de roteamento dos *hosts*

route add -net 10.10.0.0/16 dev ethX

Fonte: Elaboração da própria autora.

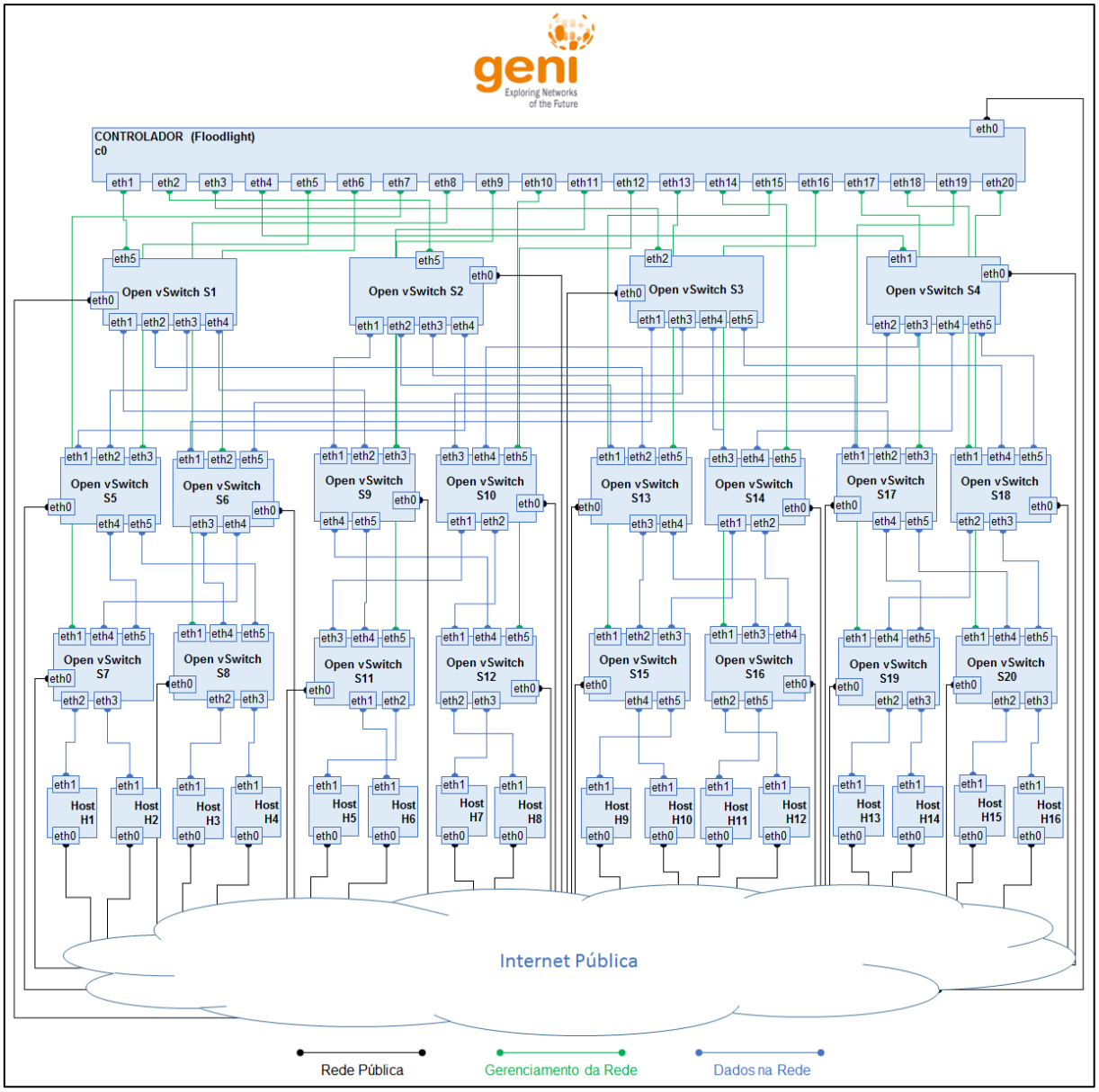
O ethX é a interface (por exemplo, eth1, eth2 em diante) de cada *host* com o IP 10.10.1.X.

Figura 18 – Topologia da fatia FatTree10 no *Rack* InstaGENI da Universidade de UtahDDC



Fonte: Elaboração da própria autora.

Figura 19 – Topologia da fatia FatTree20 no *Rack* InstaGENI da Universidade de UtahDDC



Fonte: Elaboração da própria autora.

No Apêndice J (FatTree10) e Apêndice K (FatTree20) são exibidas as configurações adicionais dos componentes de rede no *rack* InstaGENI da Universidade de UtahDDC (identificação do cliente, identificação do componente, tipo de *sliver*, nome do *host*, porta de comunicação, sistema operacional, tipo de hardware, nó virtual, IP público, endereço MAC do *switch*, interfaces, porta da máquina virtual do servidor, endereço MAC das interfaces, camada 3: endereço IP, tipo e máscara da rede).

7.1.1 Inicialização do controlador Floodlight

O Quadro 4 apresenta a execução da configuração padrão do controlador Floodlight.

Quadro 4 – Inicialização padrão do controlador Floodlight

```
java -jar floodlight.jar
```

Fonte: Floodlight (2013).

Devido às alterações nos arquivos de configuração, módulo e depuração, deve-se executar o controlador invocando estes arquivos. O Quadro 5 mostra a inicialização personalizada do controlador Floodlight. O uso de "–cp" é para informar ao controlador que ele deve usar o arquivo "ecoenergia.jar"; nele, tem-se todos arquivos para o funcionamento do Módulo Economia de Energia. O arquivo "logback.xml", por meio de "–Dlogback.configurationFile" registra mensagens de depuração em diferentes níveis no console, tais como, informação, aviso ou erro que podem ser retificadas no controlador. O arquivo "Main" do caminho "net.floodlightcontroller.core.main" específica qual é o método principal para iniciar a aplicação. E, por fim, "-cf" indica o arquivo de configuração "floodlightdefault.properties" alterado.

Quadro 5 – Inicialização personalizada do controlador Floodlight

```
java -cp ~/floodlight-1.0/target/floodlight.jar:ecoenergia.jar
-Dlogback.configurationFile=~/floodlight-1.0/logback.xml
net.floodlightcontroller.core.Main
-cf ~/floodlight-1.0/src/main/resources/floodlightdefault.properties
```

Fonte: Adaptado de Floodlight (2013).

Nas Figuras 18 e 19, a comunicação entre os *hosts* virtuais é realizada por meio do comutador por software (Open vSwitch). A máquina central (c0) possui serviços de gerenciamento da plataforma e um controlador OpenFlow (Floodlight). O OpenFlow é uma API para o controle de plataformas de encaminhamento por *software*. O encaminhamento em OpenFlow está baseado na definição de tabelas de regras para os fluxos, no qual um fluxo é um conjunto de pacotes que possui características iguais em seus campos de endereço MAC, IP de origem e destino, número de porta da camada de transporte, entre outros. Para cada fluxo é definida uma interface de saída. Em uma rede OpenFlow, os *switches* recebem essas regras de

um controlador que roda uma aplicação de controle. Assim, os operadores de redes podem instalar proativamente algumas regras padrão nos comutadores e quando chega um pacote que não possui uma regra, o pacote é enviado para o controlador que instala uma nova regra no *switch* e, então, o pacote é encaminhado.

O comando "ovs-ofctl dump-flows" exibe a tabela de regras para os fluxos nos *switches* quando um *host* emissor encaminha fluxos para o seu receptor. Para exemplificar, supondo-se que o *host* h3 (Figura 18) deseja encaminhar fluxo para o *host* h1 (Figura 18), usando o programa Iperf[14], o *host* h1 será configurado como servidor (Quadro 6) e o *host* h3 como cliente (Quadro 7), como observa-se as informações de fluxos nos *switches* s4 (Quadro 8), s5 (Quadro 9) e s6 (Quadro 10).

Quadro 6 – Configuração do *host* h1 como receptor de fluxos

ligiapre@h1:~$ iperf -s

Fonte: Adaptado de GENI (2016).

Quadro 7 – Configuração do *host* h3 como emissor de fluxos

ligiapre@h3:~$ iperf -c h1

Fonte: Adaptado de GENI (2016).

Quadro 8 – Tabela de regras para os fluxos no *switch* s4

ligiapre@s4:~$sudo ovs-ofctl dump-flows s4 NXST_FLOW reply (xid=0x4):	
Envio h3	cookie=0x20000000000000, duration=11.347s, table=0, n_packets=992, n_bytes=2702304, idle_timeout=5, idle_age=0, priority=0, in_port=2, vlan_tci=0x0000, dl_src=02:32:6f:e3:55:ff, dl_dst=02:a0:f4:bc:17:ae actions=output:1
Retorno h1	cookie=0x20000000000000, duration=11.340s, table=0, n_packets=992, n_bytes=69808, idle_timeout=5, idle_age=0, priority=0, in_port=1, vlan_tci=0x0000, dl_src=02:a0:f4:bc:17:ae, dl_dst=02:32:6f:e3:55:ff actions=output:2

Fonte: Adaptado de GENI (2016).

[14]Iperf é um software utilizado para testar a largura de banda, podendo realizar inserção de pacotes (tanto TCP quanto UDP) para medir o desempenho de redes de computadores.

Quadro 9 – Tabela de regras para os fluxos no *switch* s5

ligiapre@s5:~$sudo ovs-ofctl dump-flows s5 NXST_FLOW reply (xid=0x4):	
Envio h3	cookie=0x20000000000000, duration=12.595s, table=0, n_packets=1116, n_bytes=3069592, idle_timeout=5, idle_age=0, priority=0, in_port=3, vlan_tci=0x0000, dl_src=02:32:6f:e3:55:ff, dl_dst=02:a0:f4:bc:17:ae actions=output:2
Retorno h1	cookie=0x20000000000000, duration=12.592s, table=0, n_packets=1115, n_bytes=77918, idle_timeout=5, idle_age=0, priority=0, in_port=2, vlan_tci=0x0000, dl_src=02:a0:f4:bc:17:ae, dl_dst=02:32:6f:e3:55:ff actions=output:3

Fonte: Adaptado de GENI (2016).

Quadro 10 – Tabela de regras para os fluxos no *switch* s6

ligiapre@s6:~$sudo ovs-ofctl dump-flows s6 NXST_FLOW reply (xid=0x4):	
Envio h3	cookie=0x20000000000000, duration=13.783s, table=0, n_packets=1198, n_bytes=3312476, idle_timeout=5, idle_age=0, priority=0, in_port=1, vlan_tci=0x0000, dl_src=02:32:6f:e3:55:ff, dl_dst=02:a0:f4:bc:17:ae actions=output:3
Retorno h1	cookie=0x20000000000000, duration=13.771s, table=0, n_packets=1198, n_bytes=83404, idle_timeout=5, idle_age=0, priority=0, in_port=3, vlan_tci=0x0000, dl_src=02:a0:f4:bc:17:ae, dl_dst=02:32:6f:e3:55:ff actions=output:1

Fonte: Adaptado de GENI (2016).

Para compreender as tabelas de regras, é explanado a sequência de envio do fluxo do *host* 3 para *host* 1 e vice-versa:

a) **No *switch* s6 (Quadro 10):** envio h3 - o fluxo do *host* 3 (dl_src=02:32:6f:e3:55:ff) entrou no *switch* s6 pela porta eth1 (in_port=1) e saiu pela porta eth3 (actions=output:3);
b) **No *switch* s4 (Quadro 8):** envio h3 - o fluxo entrou no *switch* s4 pela porta eth2 (in_port=2) e saiu pela porta eth1 (actions=output:1);

c) **No *switch* s5 (Quadro 9):** envio h3 - o fluxo entrou no *switch* s5 pela porta eth3 (in_port=3) e saiu pela porta eth2 (actions=output:2). O fluxo chega ao *host* 1 (dl_dst=02:a0:f4:bc:17:ae).

Do mesmo modo, é explicada a sequência de retorno do fluxo do *host* 1 para *host* 3:

a) **No *switch* s5 (Quadro 9):** retorno h1 - o fluxo do *host* 1 (dl_src=02:a0:f4:bc:17:ae) entrou no *switch* s5 pela porta eth2 (in_port=2) e saiu pela porta eth3 (actions=output:3);
b) **No *switch* s4 (Quadro 8):** retorno h1 - o fluxo entrou no *switch s*4 pela porta eth1 (in_port=1) e saiu pela porta eth2 (actions=output:2);
d) **No *switch* s6 (Quadro 10):** retorno h1 - o fluxo entrou no *switch* s6 pela porta eth3 (in_port=3) e saiu pela porta eth1 (actions=output:1). O fluxo chega ao *host* 3 (dl_dst=02:32:6f:e3:55:ff).

7.2 RESULTADOS

Os resultados foram demonstrados por meio de um sistema de registro e realizou-se testes para depurar o comportamento do controlador. As métricas de total de pacotes, média de atraso, média da variação estatística de atraso, bytes recebidos, vazão e taxa média de pacotes foram observados nos dispositivos para constatar a atividade do Módulo Economia de Energia. Além disso, a acessibilidade dos *hosts* é verificada para conferir a exatidão da Árvore de Extensão Mínima calculada.

Os testes apresentaram três simulações em cada topologia mostrada na Figura 18 (FatTree10) e na Figura19 (FatTree20). A primeira simulação foi realizada sem o Módulo Economia de Energia no controlador Floodlight para comparar os resultados das métricas com o módulo, sendo executadas no controlador. Na segunda, com o módulo já inserido, foram avaliados os custos iniciais nas ligações entre os *switches*. Estes custos serão carregados na inicialização do controlador (Tabelas 3 e 4) e foram sugeridos para cada velocidade da porta Ethernet do *switch*.

Tabela 3 – Valores dos custos iniciais entre as conexões dos *switches* para a topologia FatTree10

Conexão entre o *switch*:	**Largura de banda**	**Valor do custo**
s1 com os *switches* s3, s4, s7 e s8	10 Mbps	10
s2 com os *switches* s3, s4, s7 e s8	10 Mbps	10
s3 com os *switches* s5 e s6	08 Mbps	08
s4 com os *switches* s5 e s6	08 Mbps	08
s7 com os *switches* s9 e s10	08 Mbps	08
s8 com os *switches* s9 e s10	08 Mbps	08

Fonte: Elaboração da própria autora.

Tabela 4 – Valores dos custos iniciais entre as conexões dos *switches* para a topologia FatTree20

Conexão entre o *switch*:	**Largura de banda**	**Valor do custo**
s1 com os *switches* s5, s9, s13 e s17	10 Mbps	10
s2 com os *switches* s5, s9, s13 e s17	10 Mbps	10
s3 com os *switches* s6, s10, s14 e s18	10 Mbps	10
s4 com os *switches* s6, s10, s14 e s18	10 Mbps	10
s5 com os *switches* s7 e s8	08 Mbps	08
s6 com os *switches* s7 e s8	08 Mbps	08
s9 com os *switches* s11 e s12	08 Mbps	08
s10 com os *switches* s11 e s12	08 Mbps	08
s13 com os *switches* s15 e s16	08 Mbps	08
s14 com os *switches* s15 e s16	08 Mbps	08
s17 com os *switches* s19 e s20	08 Mbps	08
s18 com os *switches* s19 e s20	08 Mbps	08

Fonte: Elaboração da própria autora.

Na terceira simulação, com o controlador ainda em execução, foram alterados os custos de todas as ligações (Tabelas 5 e 6) para evidenciar que o Módulo Economia de Energia é capaz de calcular uma nova Árvore de Extensão Mínima sobre os custos fornecidos e assim adaptar-se à rede para uma nova circunstância de atualização.

Tabela 5 – Valores de novos custos entre as conexões dos *switches* para a topologia FatTree10

Conexão entre o *switch*:	**Largura de banda**	**Valor do custo**
s1 com os *switches* s3, s4, s7 e s8	10 Mbps	16
s2 com os *switches* s3, s4, s7 e s8	10 Mbps	14
s3 com os *switches* s5 e s6	08 Mbps	12
s4 com os *switches* s5 e s6	08 Mbps	12
s7 com os *switches* s9 e s10	08 Mbps	10
s8 com os *switches* s9 e s10	08 Mbps	10

Fonte: Elaboração da própria autora.

Tabela 6 – Valores de novos custos entre as conexões dos *switches* para a topologia FatTree20

Conexão entre o *switch*:	**Largura de banda**	**Valor do custo**
s1 com os *switches* s5, s9, s13 e s17	10 Mbps	18
s2 com os *switches* s5, s9, s13 e s17	10 Mbps	14
s3 com os *switches* s6, s10, s14 e s18	10 Mbps	16
s4 com os *switches* s6, s10, s14 e s18	10 Mbps	12
s5 com os *switches* s7 e s8	08 Mbps	16
s6 com os *switches* s7 e s8	08 Mbps	16
s9 com os *switches* s11 e s12	08 Mbps	12
s10 com os *switches* s11 e s12	08 Mbps	12
s13 com os *switches* s15 e s16	08 Mbps	14
s14 com os *switches* s15 e s16	08 Mbps	14
s17 com os *switches* s19 e s20	08 Mbps	10
s18 com os *switches* s19 e s20	08 Mbps	10

Fonte: Elaboração da própria autora.

7.2.1 Configuração do tráfego de dados gerado pelo D-ITG usado nas três simulações

Durante os três experimentos foram realizados testes de conectividade entre os *hosts* e emissão de fluxos de dados gerados pelo D-ITG, exibido no Apêndice I.

O *host* h1 será escolhido para ser o receptor dos dados enviados aos outros *hosts* emissores da topologia FatTree10 (h2, h3, h4, h5, h6, h7 e h8) e o mesmo ocorre para o *host* h1 da topologia FatTree20 (h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, h13, h14, h15 e h16).

A fim de executar os testes, serão criados fluxos para emissão de dados de cada *host* remetente. A Tabela 7 apresenta as configurações do tráfego de dados.

Tabela 7 – Configuração do tráfego de dados para as simulações

Descrição	**Parâmetros**
Quantidade de fluxos	10
Protocolo empregado	TCP
Tamanho de cada pacote enviado	512 bytes
Taxa de transferência constante	100 pps (pacotes por segundo)
Duração dos fluxos	30 s (segundos)

Fonte: Elaboração da própria autora.

O *host* h1 não será mostrado nos gráficos, uma vez que os seus valores serão semelhantes aos exibidos nos demais *hosts*.

Considerando a configuração do tráfego de dados apresentado na Tabela 7 e a definição da largura de banda em 2 Mbps nas conexões entre os *switches* de borda e os *hosts*, estima-se no desempenho da rede os valores aproximados para cada *host*:

a) **Total de pacotes:**10 fluxos * 100 pps * 30s = 30.000 pacotes. Sendo a largura de banda em 2 Mbps será em torno de 14.648 pacotes;
b) **Bytes recebidos:** 10 fluxos * 512 bytes * 100 pps * 30s = 15.360.000 bytes. Consistindo a largura de banda em 2 Mbps ocorrerá em média de 7.500.000 bytes;
c) **Taxa média de pacotes:** Total de pacotes / 30s = 1.000 pacotes por segundo. Compondo a largura de banda em 2 Mbps corresponderá aproximadamente 488 pacotes por segundo;
d) **Média de atraso:** o atraso é medido entre a diferença do tempo de recepção (rxTime) e o tempo de transmissão (txTime) de cada pacote. O valor médio de atraso deve ser próximo a zero;
e) **Média da variação estatística de atraso:** a variação estatística de atraso é computada entre a variação da diferença do tempo de recepção (rxTime) e o tempo de transmissão (txTime) de cada pacote. O valor médio da variação estatística de atraso deve ser próximo a zero;

f) **Vazão:** (((512 bytes * 10 fluxos) * 8 bits por byte) * 100 pps) / 1000 = 4.096 kilobits por segundo. Tratando-se a largura de banda em 2 Mbps ficará em cerca de 2.000 kilobits por segundo.

7.2.2 Primeira simulação: sem o Módulo Economia de Energia

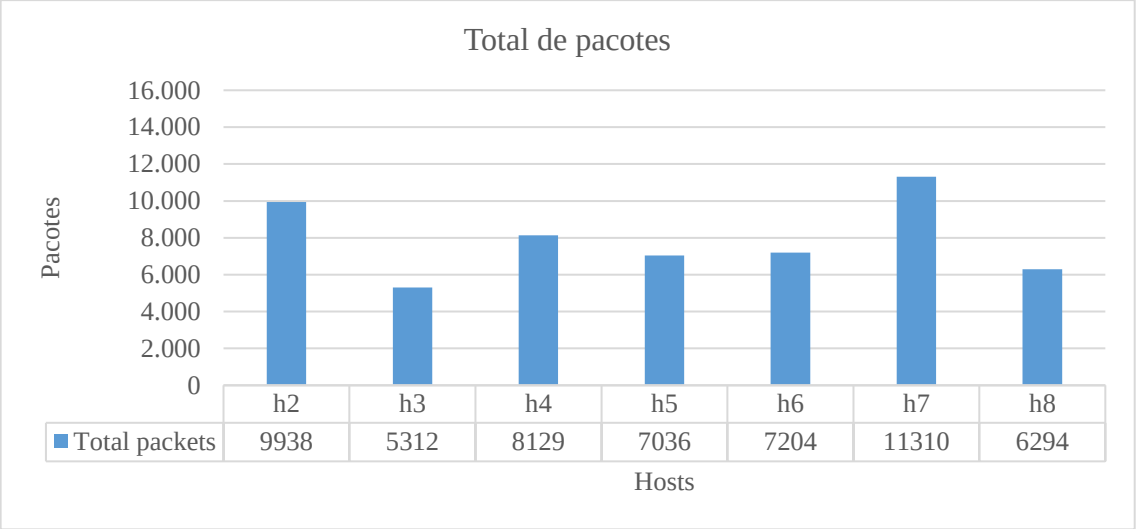
Nesta simulação será analisado o comportamento do controlador Floodlight na sua configuração padrão, ou seja, com os módulos predefinidos, exibido na Figura 12.

7.2.2.1 Análise de desempenho da rede com o tráfego de dados gerado pelo D-ITG

Considerando os registros das simulações realizadas no dia vinte e oito de agosto de dois mil e dezesseis (28/08/2016), iniciou-se a execução do controlador Floodlight (Quadro 4) às 15h00min30s400ms para a topologia FatTree10 e às 19h00min40s555ms para a topologia FatTree20. Em ambas topologias foi observado com o comando “ovs-ofctl show” que todas as portas Ethernet dos *switches* se encontravam ligadas aguardando o tráfego na rede. A simulação foi finalizada para a topologia FatTree10 as 16h00min30s900ms e para a topologia FatTree20 as 20h00min25s900ms. A configuração do tráfego de dados usada nas duas topologias foi explanada na subseção 7.2.1.

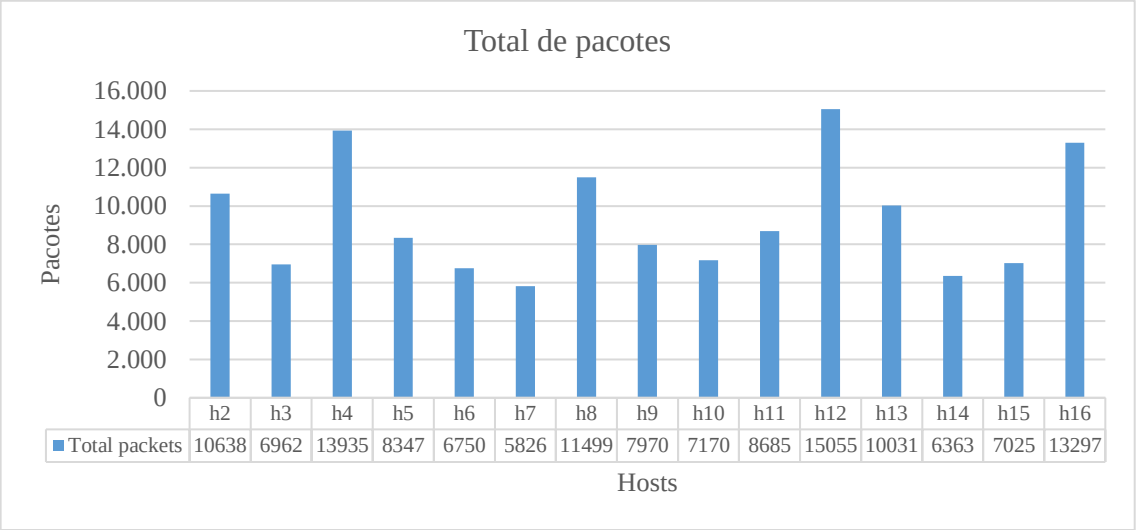
Nas métricas apresentadas, total de pacotes, bytes recebidos e taxa média de pacotes, o fluxo de dados é transmitido pelos *hosts* ao receptor h1 e vice-versa, em ambas topologias. É verificado o total de pacotes em cada topologia apresentada nas Figuras 20 e 21. Observa-se que no período das 15h00min às 16h00min, a média de pacotes para a topologia FatTree10 foi de 7.889 pacotes e no período das 19h00min às 20h00min, para a topologia FatTree20 foi de 9.303 pacotes. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento adequado.

Figura 20 – Total de pacotes entre os *hosts* para a topologia FatTree10



Fonte: Elaboração da própria autora.

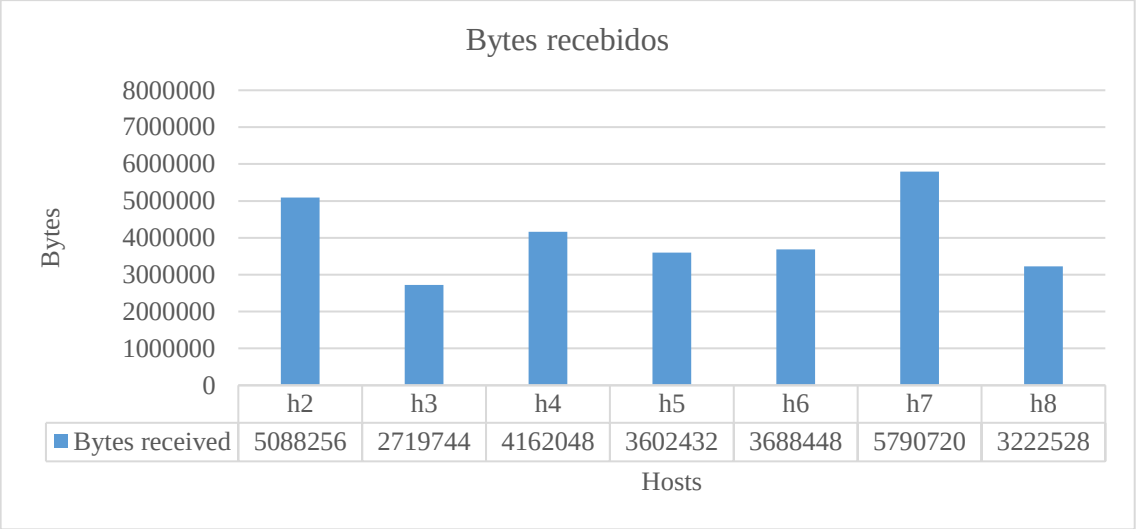
Figura 21 – Total de pacotes entre os *hosts* para a topologia FatTree20



Fonte: Elaboração da própria autora.

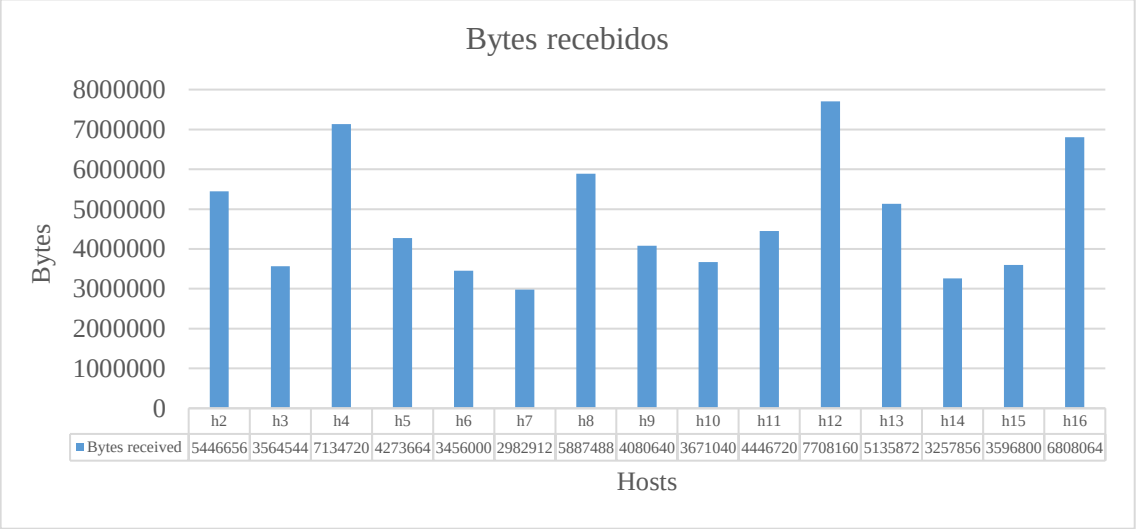
Do mesmo modo, é apurada a quantidade de bytes recebidos em cada topologia apresentada nas Figuras 22 e 23. Observa-se que no período das 15h00min às 16h00min, a média de bytes convertido em megabytes para a topologia FatTree10 foi de 4,03 MB e no período das 19h00min às 20h00min para a topologia FatTree20 foi de 4,76 MB. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento pertinente.

Figura 22 – Bytes recebidos entre os *hosts* para a topologia FatTree10



Fonte: Elaboração da própria autora.

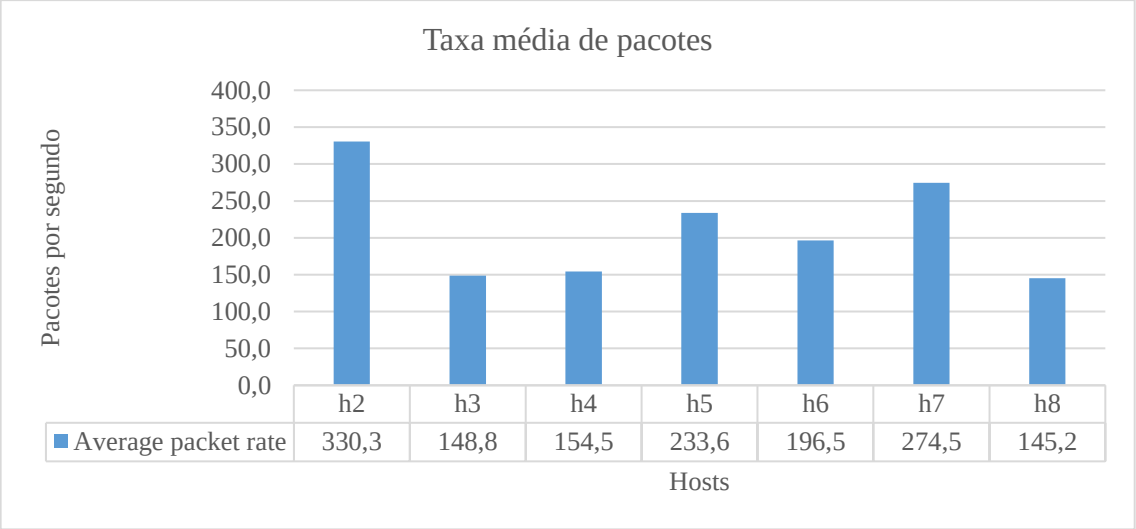
Figura 23 – Bytes recebidos entre os *hosts* para a topologia FatTree20



Fonte: Elaboração da própria autora.

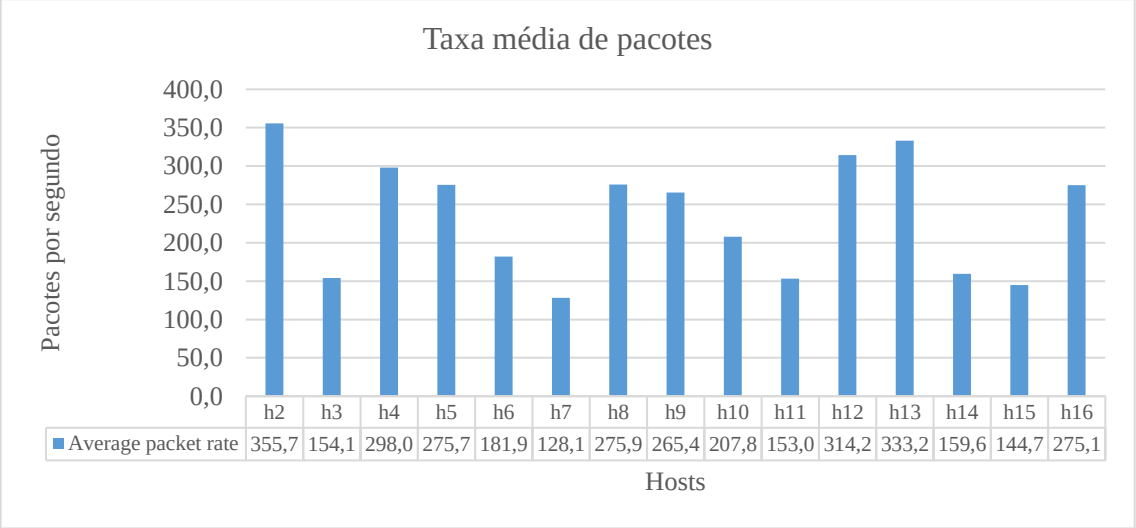
As Figuras 24 e 25 apresentam a taxa média de pacotes por segundo em cada topologia. Observa-se que no período das 15h00min às 16h00min, a média da taxa de pacotes foi de 211,9 pacotes por segundo para a topologia FatTree10 e, no período das 19h00min às 20h00min, para a topologia FatTree20, foi de 234,8 pacotes por segundo. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento efetivo.

Figura 24 – Taxa média de pacotes entre os *hosts* para a topologia FatTree10



Fonte: Elaboração da própria autora.

Figura 25 – Taxa média de pacotes entre os *hosts* para a topologia FatTree20



Fonte: Elaboração da própria autora.

Observando os valores esperados da subseção 7.2.1 em relação ao desempenho da rede, os resultados são considerados satisfatórios. Alguns *hosts* apresentaram valores abaixo do esperado, devido ao congestionamento e atraso na transmissão de dados.

7.2.3 Segunda simulação: com o módulo economia de energia e custos iniciais nas ligações entre os *switches*

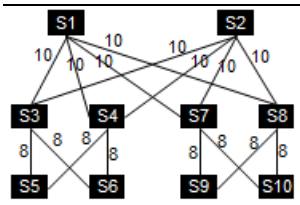
Nesta segunda simulação será analisado o comportamento do controlador Floodlight com a inclusão do Módulo Economia de Energia incluindo os custos iniciais nas ligações entre os *switches*. Os valores dos custos correspondem à largura de banda de cada ligação apresentados nas Tabelas 3 (FatTree10) e 4 (FatTree20).

7.2.3.1 Investigação do cálculo da árvore de extensão mínima pelo algoritmo MiNet

Após a inicialização do controlador (Quadro 5), como exemplo, foi realizado a investigação na topologia FatTree10 em relação ao algoritmo MiNet que calcula a Árvore de Extensão Mínima até encontrar uma solução para a conectividade de todos os *switches*, sem a formação de ciclos que fecham os seus vértices e, assim, economizando energia, desativa-se algumas portas Ethernet ociosas.

O grafo original da rede é mostrado na Tabela 8, os números próximos das ligações significam o seu custo. A escolha do *switch* pode ser feita randomicamente ou pelo administrador de rede.

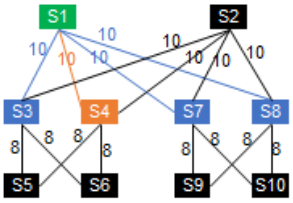
Tabela 8 – Grafo original da rede

Grafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{}	{}	{}	{s1, s2, s3, s4 s5, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

O *switch* s1 foi escolhido como ponto inicial do algoritmo. Os *switches* s3, s4, s7 e s8 estão conectados com s1 por meio de uma única ligação. Esses *switches* apresentam o mesmo custo (10) em suas ligações, então, escolhe-se qualquer ligação. O *switch* s4 foi selecionado e, portanto, a ligação {s1s4} será indicada para formar o subgrafo, apresentada na Tabela 9.

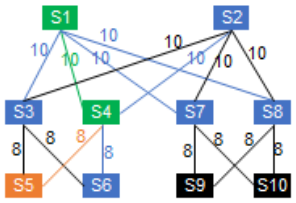
Tabela 9 – Subgrafo da rede: ligação {s1s4} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1}	{s1,s3}=10 {s1,s4}=10 {s1,s7}=10 {s1,s8}=10	{s1s4}	{s2, s3, s4, s5, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

O próximo *switch* escolhido é o mais perto de s1 ou s4. Os *switches* s3, s7 e s8 estão a uma distância 10 de s1. S2 está numa distância 10 de s4. S5 e s6 estão a uma distância 8 de s4. Os *switches* s5 e s6 são os mais próximos (menor custo); elege-se qualquer uma dessas ligações. Logo, será escolhida a ligação {s4s5}, mostrada na Tabela 10.

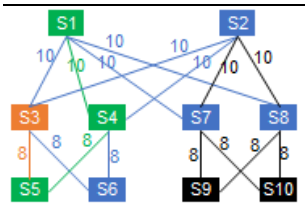
Tabela 10 – Subgrafo da rede: ligação {s4s5} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s4}	{s1,s3}=10 {s1,s7}=10 {s1,s8}=10 {s4,s2}=10 {s4,s5}=8 {s4,s6}=8	{s1s4, s4s5}	{s2, s3, s5, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

Agora, deve-se escolher o *switch* mais próximo de s1, s4 ou s5. S3, s7 e s8 estão a uma distância 10 de s1. S2 está numa distância 10 e s6 numa distância 8 de s4. S3 está a uma distância 8 de s5. Os *switches* s3 e s6 são os mais próximos (menor custo) respectivamente de s5 e s4; seleciona-se qualquer uma dessas ligações. A seguir, será escolhida a ligação {s5s3}, mostrada na Tabela 11.

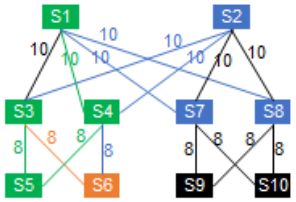
Tabela 11 – Subgrafo da rede: ligação {s5s3} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s4, s5}	{s1,s3}=10 {s1,s7}=10 {s1,s8}=10 {s4,s2}=10 {s4,s6}=8 {s5,s3}=8	{s1s4, s4s5, s5s3}	{s2, s3, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

A subsequente escolha está entre os *switches* s1, s4 e s3. S3, s7 e s8 estão a uma distância 10 de s1. S2 está numa distância 10 de s4 e s3. S6 está numa distância 8 de s4 e s3. A ligação {s1s3} está fora do subgrafo, devido à formação de ciclo. O *switch* s6 é o mais próximo (menor custo) de s3 e s4; seleciona-se qualquer uma dessas ligações. Então, será designada a ligação {s3s6}, mostrada na Tabela 12.

Tabela 12 – Subgrafo da rede: ligação {s3s6} selecionada

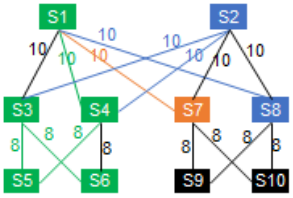
Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5}	{s1,s3}=10 ciclo {s1,s7}=10 {s1,s8}=10 {s4,s2}=10 {s4,s6}=8 {s3,s2}=10 {s3,s6}=8	{s1s4, s4s5, s5s3, s3,s6}	{s2, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora

Neste momento, os *switches* s1, s3 e s4 estão disponíveis. A ligação de {s4s6} está fora do subgrafo, devido à formação de ciclo. S7 e s8 estão a uma distância 10 de s1. S2 está numa

distância 10 de s4 e s3. Escolhe-se qualquer uma dessas ligações. Portanto, a ligação {s1s7} será nomeada e apresentada na Tabela 13.

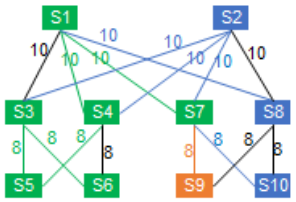
Tabela 13 – Subgrafo da rede: ligação {s1s7} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5, s6}	{s1,s3}=10 ciclo {s1,s7}=10 {s1,s8}=10 {s4,s2}=10 {s4,s6}=8 ciclo {s3,s2}=10	{s1s4, s4s5, s5s3, s3,s6, s1s7}	{s2, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

Os *switches* s1, s3 e s4 continuam no subgrafo. S7 foi adicionado. S8 está a uma distância de 10 de s1. S2 está a uma distância 10 de s3, s4 e s7. S9 e s10 estão em uma distância 8 de s7. Os *switches* s9 e s10 são os mais próximos (menor custo) de s7; seleciona-se qualquer uma dessas ligações. Assim, será escolhida a ligação {s7s9}, mostrada na Tabela 14.

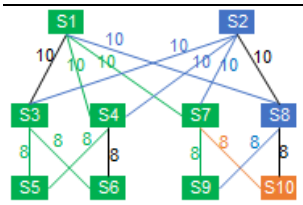
Tabela 14 – Subgrafo da rede: ligação {s7s9} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5, s6, s7}	{s1,s3}=10 ciclo {s1,s8}=10 {s4,s2}=10 {s4,s6}=8 ciclo {s3,s2}=10 {s7,s2}=10 {s7,s9}=8 {s7,s10}=8	{s1s4, s4s5, s5s3, s3,s6, s1s7, s7s9}	{s2, s8, s9, s10}

Fonte: Elaborado pelo autor.

A escolha está entre os *switches* s1, s3, s4, s7 e s9. S8 está a uma distância 10 de s1 e numa distância 8 de s9. S2 está a uma distância de 10 de s3, s4 e s7. S10 está em uma distância 8 de s7. Os *switches* s8 e s10 são os mais próximos (menor custo) respectivamente de s9 e s7; escolhe-se qualquer uma dessas ligações. Desse modo, será escolhida a ligação {s7s10}, mostrada na Tabela 15.

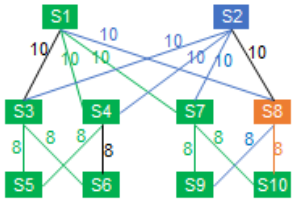
Tabela 15 – Subgrafo da rede: ligação {s7s10} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5, s6, s7, s9}	{s1,s3}=10 ciclo {s1,s8}=10 {s4,s2}=10 {s4,s6}=8 ciclo {s3,s2}=10 {s7,s2}=10 {s7,s10}=8 {s9,s8}=8	{s1s4, s4s5, s5s3, s3,s6, s1s7, s7s9, s7s10}	{s2, s8, s10}

Fonte: Elaboração da própria autora.

Os *switches* s1, s3 s4, s7, s9 e s10 estão no subgrafo. S8 está a uma distância 10 de s1 e numa distância 8 de s9 e s10. S2 está a uma distância 10 de s3, s4 e s7. O *switch* s8 é o mais próximo (menor custo) de s9 e s10; elege-se qualquer uma dessas ligações. Logo, será escolhida a ligação {s10s8}, mostrada na Tabela 16.

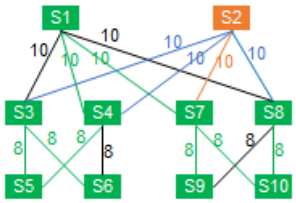
Tabela 16 – Subgrafo da rede: ligação {s10s8} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5, s6, s7, s9, s10}	{s1,s3}=10 ciclo {s1,s8}=10 {s4,s2}=10 {s4,s6}=8 ciclo {s3,s2}=10 {s7,s2}=10 {s9,s8}=8 {s10,s8}=8	{s1s4, s4s5, s5s3, s3,s6, s1s7, s7s9, s7s10, s10s8}	{s2, s8}

Fonte: Elaboração da própria autora.

As opções dos *switches* s3, s4, s7 e s8 estão no subgrafo. As ligações {s1s8} e {s9s8} estão fora do subgrafo, devido a formação de ciclo. S2 está a uma distância 10 de s3, s4, s7 e s8. Neste caso escolhe-se qualquer um. Portanto, é selecionada a ligação {s7s2}, mostrada na Tabela 17.

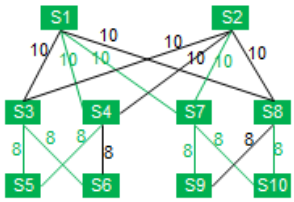
Tabela 17 – Subgrafo da rede: ligação {s7s2} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5, s6, s7, s8, s9, s10}	{s1,s3}=10 ciclo {s1,s8}=10 ciclo {s4,s2}=10 {s4,s6}=8 ciclo {s3,s2}=10 {s7,s2}=10 {s9,s8}=8 ciclo {s8,s2}=10	{s1s4, s4s5, s5s3, s3,s6, s1s7, s7s9, s7s10, s10s8, s7s2}	{s2}

Fonte: Elaboração da própria autora.

As ligações {s4s2}, {s3s2} e {s8s2} estão fora do subgrafo, devido à formação de ciclos. Aqui está o fim da investigação do subgrafo formado pelas ligações em verde que representam a Árvore de Extensão Mínima, mostrado na Tabela 18. Nesse caso, a árvore apresenta a soma de todas as suas ligações o valor 78.

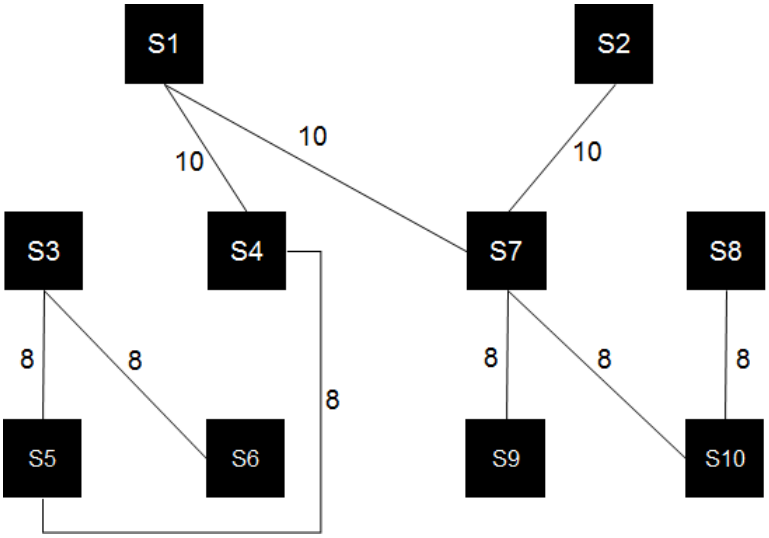
Tabela 18 – Resultado do subgrafo da rede (custos iniciais) com o algoritmo MiNet

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s2, s3, s4, s5, s6, s7, s8, s9, s10}	{s1,s3}=10 ciclo {s1,s8}=10 ciclo {s4,s2}=10 ciclo {s4,s6}=8 ciclo {s3,s2}=10 ciclo {s9,s8}=8 ciclo {s8,s2}=10 ciclo	{s1s4, s4s5, s5s3, s3,s6, s1s7, s7s9, s7s10, s10s8, s7s2}	{}

Fonte: Elaboração da própria autora.

Com essa investigação, é possível evidenciar que o algoritmo MiNet identifica adequadamente a Árvore de Extensão Mínima, mantendo as portas ativas das ligações de baixo custo, mostrado na Figura 26, otimizando todas as conexões de *switch* a *switch* e livre de ciclos que fecham os vértices dos *switches* na rede OpenFlow.

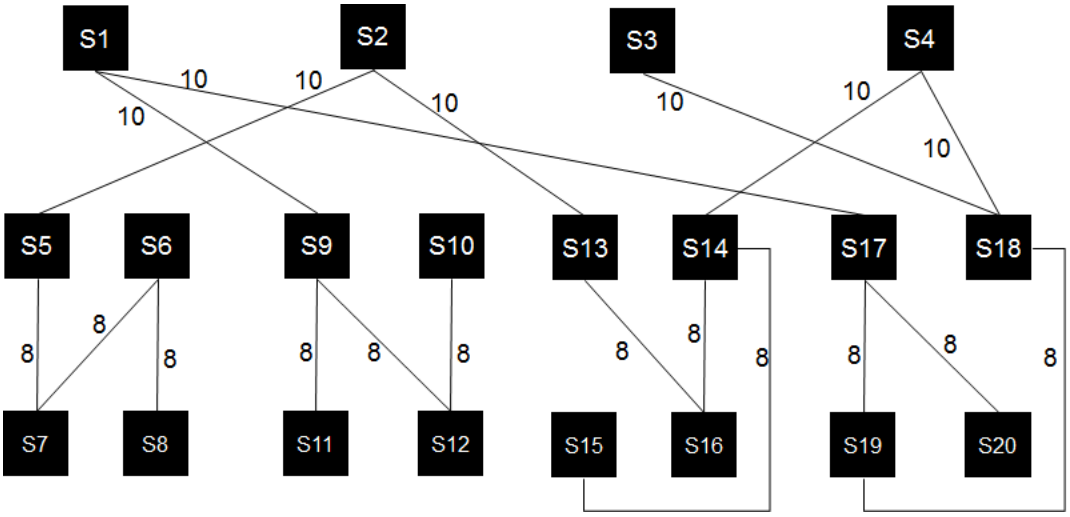
Figura 26 – Árvore de Extensão Mínima na topologia FatTree10 (custos iniciais)



Fonte: Elaboração da própria autora.

Assim sendo, realizando a investigação do cálculo da Árvore de Extensão Mínima pelo algoritmo MiNet na topologia FatTree20, será apresentado o resultado do grafo da rede na Figura 27.

Figura 27 – Árvore de Extensão Mínima na topologia FatTree20 (custos iniciais)

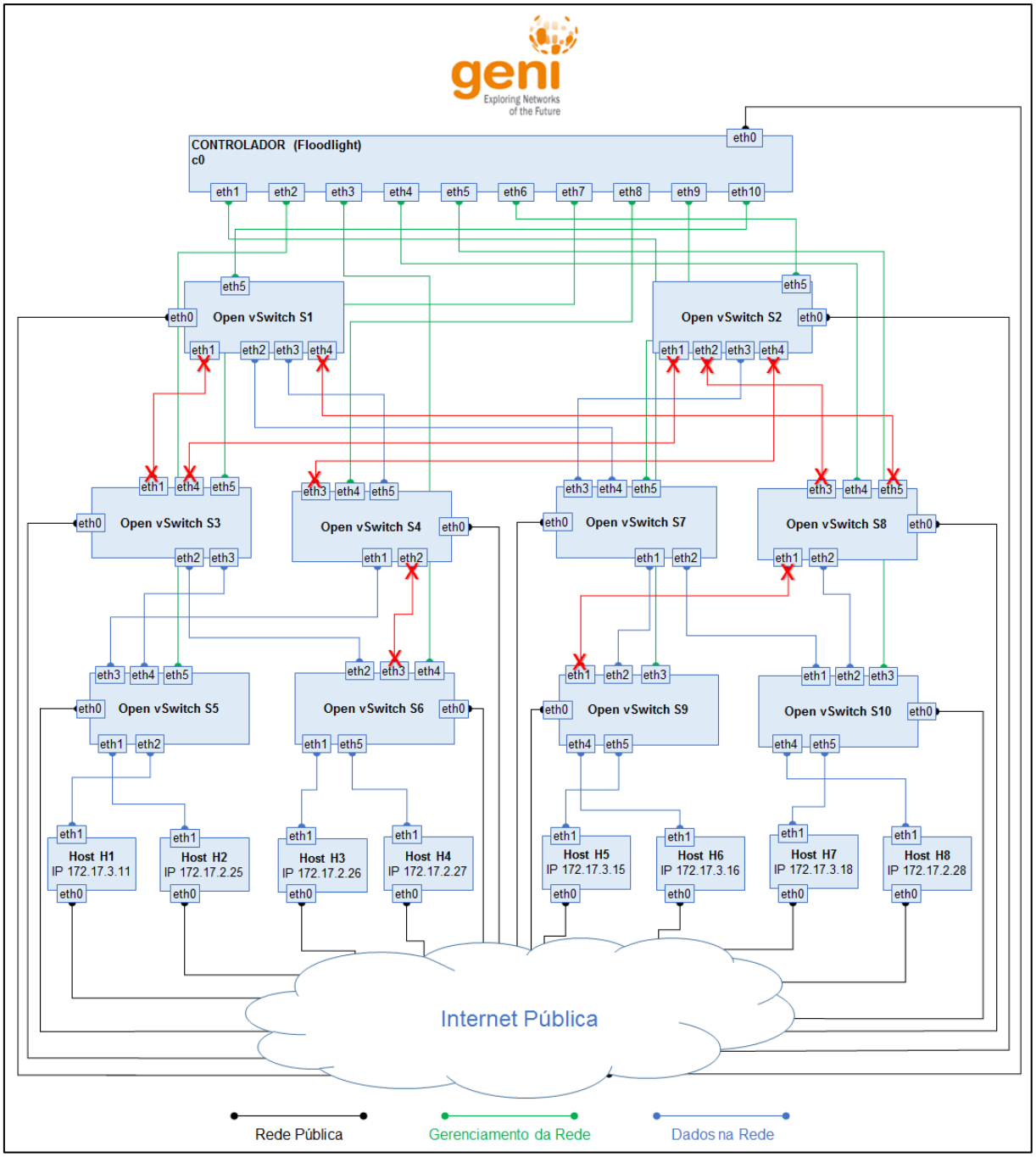


Fonte: Elaboração da própria autora.

Destaca-se que os valores dos custos foram iguais aos valores da largura de banda de cada ligação entre os *switches* e a escolha do melhor caminho (menor custo) foi opcional para as conexões com o mesmo valor em cada camada da topologia FatTree.

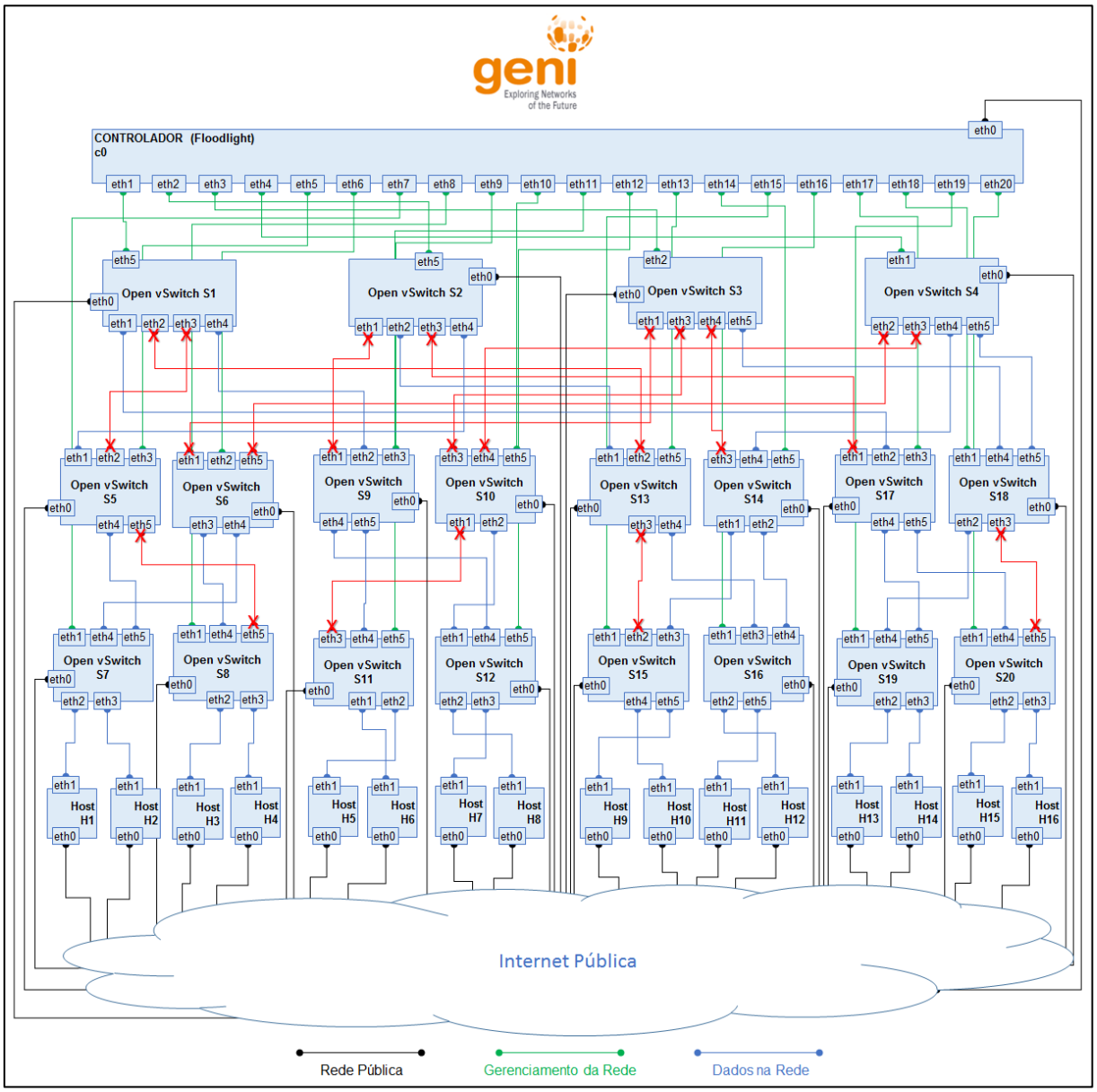
Analisando detalhadamente com o comando “ovs-ofctl show”, pôde-se observar quais portas ficaram ativadas e quais foram desativadas nas topologias propostas exibidas nas Figuras 28 e 29.

Figura 28 – Topologia FatTree10 otimizada com o Módulo Economia de Energia (custos iniciais)



Fonte: Elaboração da própria autora.

Figura 29 – Topologia FatTree20 otimizada com o Módulo Economia de Energia (custos iniciais)



Fonte: Elaboração da própria autora.

O Quadro 11 apresenta o resultado desta análise no *switch* s2 da topologia FatTree10 (Figura 28), como exemplo. A porta s2-eth1 está conectada com s3-eth4, em “config” e “state”, em configuração e estado da porta apresentam respectivamente PORT_DOWN e LINK_DOWN, significa que estão desativados. A porta s2-eth2 está conectada com s8-eth3, em configuração e estado da porta apresentam desativados. A porta s2-eth3 está conectada com s7-eth3, em configuração e estado da porta mostram o valor zero, significa, respectivamente que a porta e a ligação estão ativadas. A porta s2-eth4 está conectada com s4-eth3, em

configuração e estado da porta mostram desativadas. A porta s2-eth5 corresponde à ligação com o controlador e a porta s2-eth0 está conectado com a Internet pública. As portas s2-eth0 e s2-eth5 estão fora do cálculo da Árvore de Extensão Mínima.

Quadro 11 – Resultado das configurações e estado das portas do *switch* s2 da topologia FatTree10 (custos iniciais)

```
ligiapre@s2:~$ sudo ovs-ofctl show s2
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000002
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS
ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_VLAN SET_DL_SRC
SET_DL_DST SET_NW_SRC SET_NW_DST SET_NW_TOS SET_TP_SRC
SET_TP_DST ENQUEUE
1(eth1): addr:02:c9:4e:32:e9:9d
config:         PORT_DOWN
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
 2(eth2): addr:02:1b:b1:03:ae:54
    config:     PORT_DOWN
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
 3(eth3): addr:02:b4:2f:b2:a7:a1
    config:     0
    state:      0
    speed: 0 Mbps now, 0 Mbps max
 4(eth4): addr:02:8b:71:16:2e:84
    config:     PORT_DOWN
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
 LOCAL(s2): addr:e6:b7:33:39:f6:41
    config:     0
    state:      0
    speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Fonte: Adaptado de GENI (2016).

Para a topologia FatTree10, considerando somente as ligações entre os *switches*, do total de quarenta portas Ethernet (100%) a sobreposição da Árvore de Extensão Mínima desativaria

quatorze portas Ethernet dos *switches* (redução de 35%), sendo, s1-eth1, s1-eth4, s2-eth1, s2-eth2, s2-eth4, s3-eth1, s3-eth4, s4-eth2, s4-eth3, s6-eth3, s8-eth1, s8-eth3, s8-eth5, s9-eth1. O consumo de energia total das portas Ethernet seria de 65%.

Já para a topologia FatTree20, considerando somente as ligações entre os *switches*, do total de oitenta portas Ethernet (100%) a sobreposição da Árvore de Extensão Mínima desativaria vinte e seis portas Ethernet dos *switches* (redução de 32,5%), sendo, s1-eth2, s1-eth3, s2-eth1, s2-eth3, s3-eth1, s3-eth3, s3-eth4, s4-eth2, s4-eth3, s5-eth2, s5-eth5, s6-eth1, s6-eth5, s8-eth5, s9-eth1, s10-eth1, s10-eth3, s10-eth4, s11-eth3, s13-eth2, s13-eth3, s14-eth3, s15-eth2, s17-eth1, s18-eth3, s20-eth5. O consumo de energia total das portas Ethernet seria de 67,5%.

7.2.3.2 Avaliação do consumo de energia durante a execução da topologia proposta

Observando os registros da simulação realizada na topologia FatTree10 no dia trinta e um de agosto de dois mil e dezesseis (31/08/2016), iniciou-se a execução do controlador Floodlight (Quadro 5) às 09h02min18s792ms. O cálculo da Árvore de Extensão Mínima terminou às 09h02min33s432ms. O Módulo Economia de Energia levou quinze segundos para preparar a árvore sobreposta de conectividade entre os *switches*. A partir deste período foi inserido tráfego na rede e não observou mudança na topologia em relação às portas desativadas (Figura 28). Portanto, o consumo de energia total das portas Ethernet para esse período foi de 65% (redução de 35%), considerado satisfatório para economia de energia na rede. Os testes continuarão na terceira simulação com a inclusão de novos custos nas ligações entre os *switches* (realizado às 09h30min03s283ms) e logicamente, essa porcentagem poderá variar até o término da simulação, devido aos fluxos de dados na rede e as mudanças nos custos das ligações dos *switches*.

Analisando os registros da simulação sucedida na topologia FatTree20 no dia trinta e um de agosto de dois mil e dezesseis (31/08/2016), estabeleceu-se a execução do controlador Floodlight (Quadro 5) as 17h34min53s203ms. O cálculo da Árvore de Extensão Mínima encerrou-se às 17h35min13s664ms. O Módulo Economia de Energia levou vinte segundos para preparar a árvore sobreposta de conectividade entre os *switches*. Até as 17h37min20s682ms, a Árvore de Extensão Mínima permaneceu com as portas desativadas conforme a Figura 29, mantendo o consumo de energia total das portas Ethernet em 67,5% (redução de 32,5%). Devido ao fluxo de dados na rede, às 17h37min20s683ms teve a observação na mudança da topologia em relação às portas desativadas. As portas Ethernet s6-eth1 e s6-eth5 foram ligadas

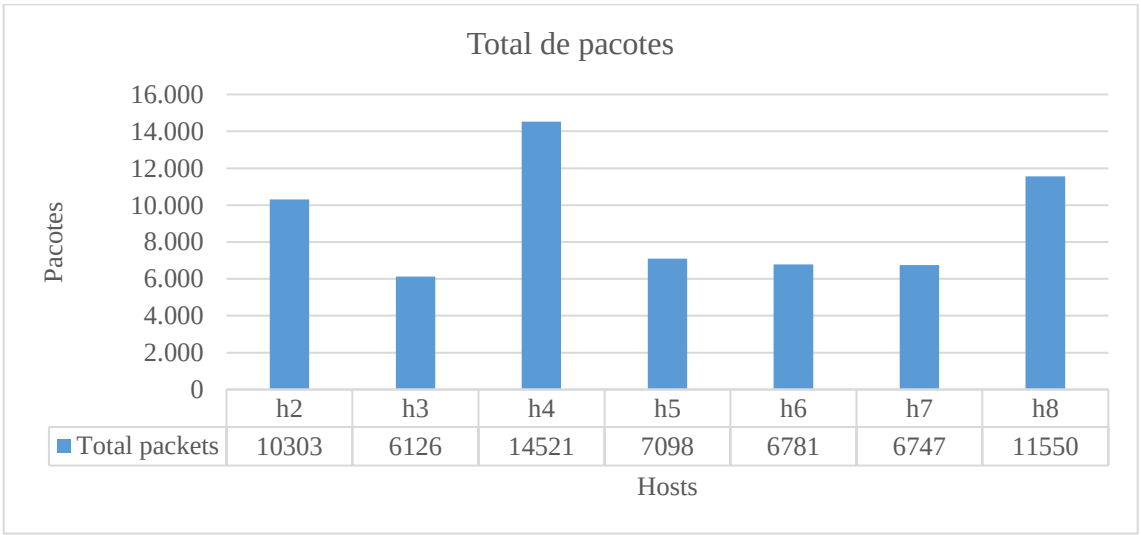
e as demais permaneceram desligadas. Por fim, do total de oitenta portas Ethernet (100%) na sobreposição da Árvore de Extensão Mínima ficaram vinte e quatro portas Ethernet dos *switches* desativadas (redução de 30%). O consumo de energia total das portas Ethernet foi de 70%, considerado satisfatório para economia de energia na rede. Os testes seguirão na terceira simulação com a inclusão de novos custos nas ligações entre os *switches* (realizado às 18h00min25s283ms) e obviamente que esta porcentagem poderá modificar até o fim da simulação, devido aos fluxos de dados na rede e as alterações nos custos das ligações entre os *switches*.

7.2.3.3 Análise de desempenho da rede sobre o tráfego de dados gerado pelo D-ITG

No decorrer da simulação de ambas topologias foram empregadas as configurações do tráfego de dados explanada na subseção 7.2.1. Em seguida, serão apresentadas as métricas adotadas (total de pacotes, bytes recebidos e taxa média de pacotes) para mensurar o desempenho do Módulo Economia de Energia com custos iniciais nas ligações dos *switches* (Apêndice B). O fluxo de dados é transmitido pelos *hosts* ao receptor h1 e vice-versa nas duas topologias.

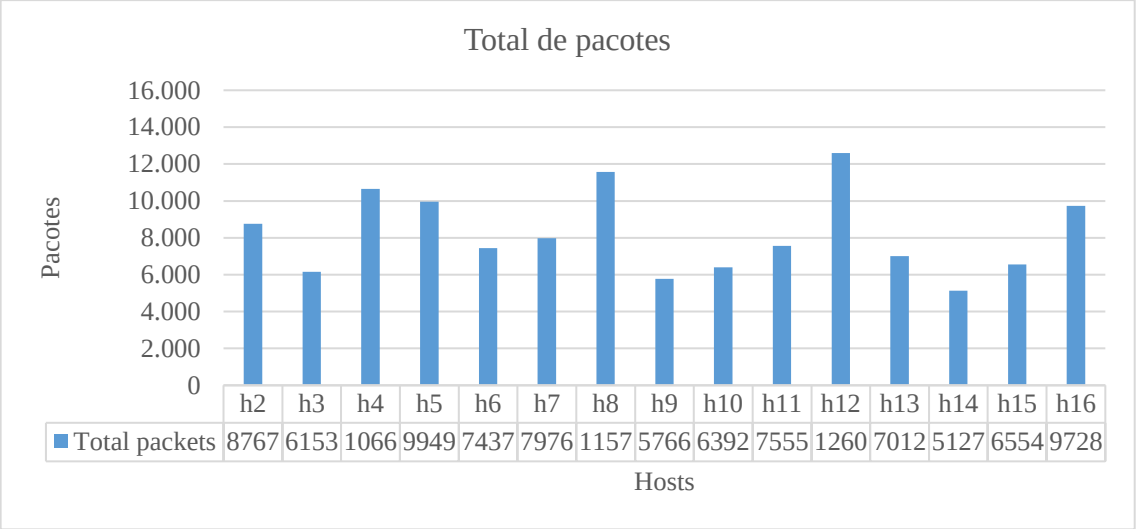
É apurado o total de pacotes em cada topologia apresentado nas Figuras 30 e 31. Observa-se que no período das 09h02min às 09h30min, a média de pacotes para a topologia FatTree10 foi de 9.018 pacotes; e no período das 17h34min às 18h00min, para a topologia FatTree20 foi de 8.217 pacotes. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento apropriado.

Figura 30 – Total de pacotes entre os *hosts* na topologia FatTree10 (custos iniciais)



Fonte: Elaboração da própria autora.

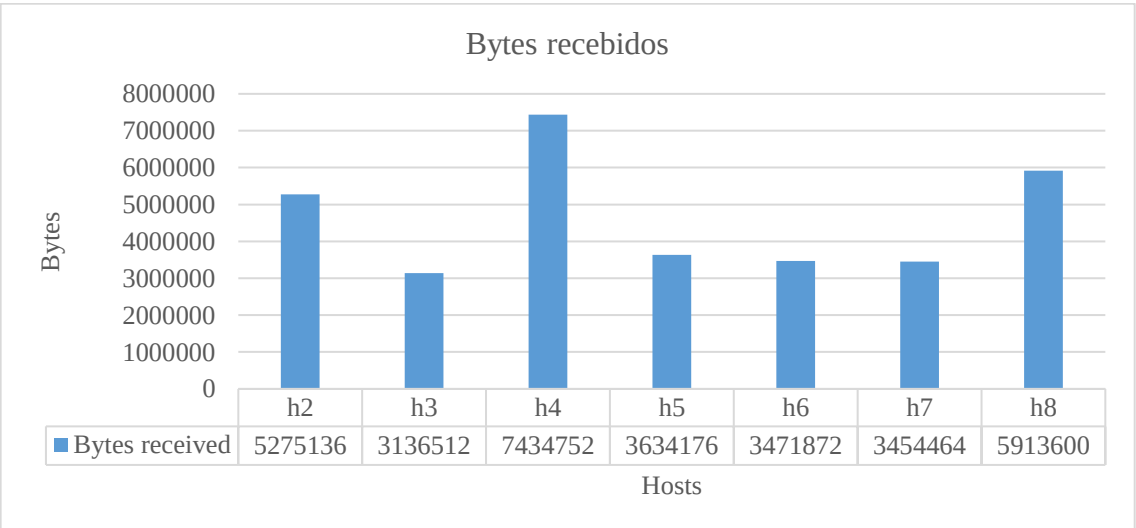
Figura 31 – Total de pacotes entre os *hosts* na topologia FatTree20 (custos iniciais)



Fonte: Elaboração da própria autora.

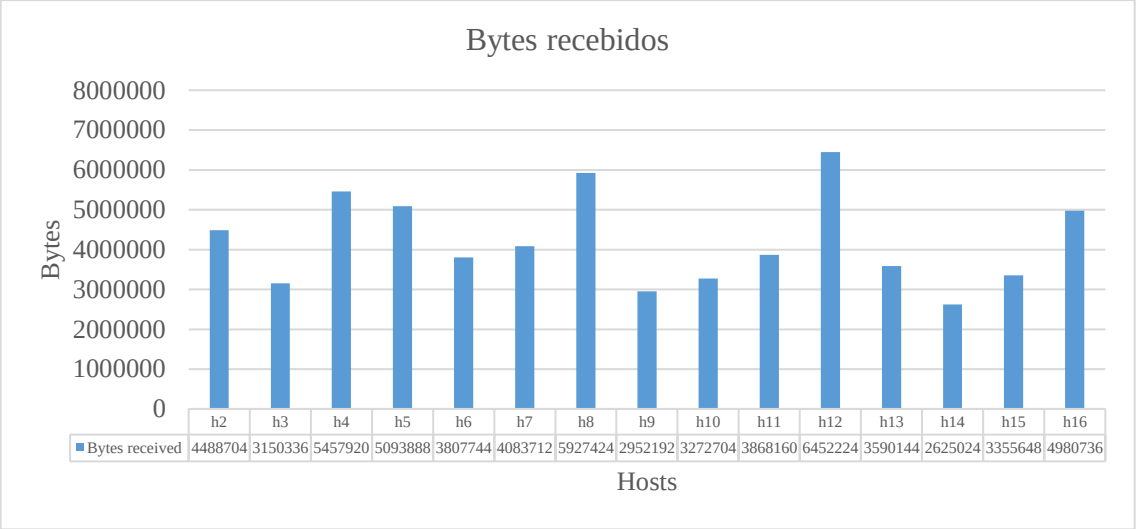
Assim como nas Figuras 32 e 33 são apurados a quantidade de bytes recebidos em cada topologia, observa-se que no período das 09h02min às 09h30min, a média de bytes convertido em megabytes para a topologia FatTree10 foi de 4,61 MB; e no período das 17h34min às 18h00min para a topologia FatTree20 foi de 4,20 MB. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento pertinente.

Figura 32 – Bytes recebidos entre os *hosts* na topologia FatTree10 (custos iniciais)



Fonte: Elaboração da própria autora.

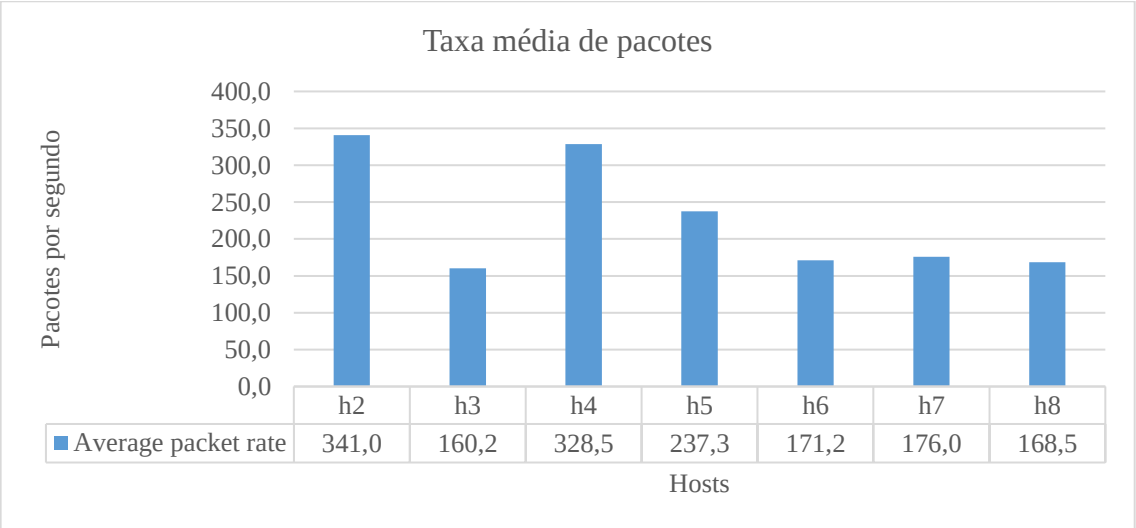
Figura 33 – Bytes recebidos entre os *hosts* na topologia FatTree20 (custos iniciais)



Fonte: Elaboração da própria autora.

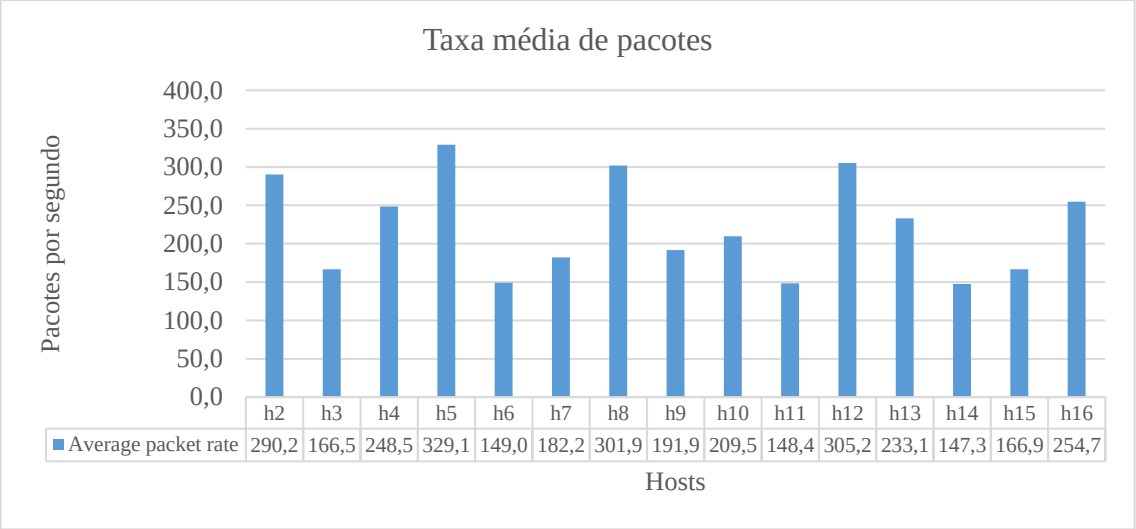
As Figuras 34 e 35 apresentam a taxa média de pacotes por segundo em cada topologia. Nota-se que no período das 09h02min às 09h30min, a média da taxa de pacotes foi de 226,1 pacotes por segundo para a topologia FatTree10 e no período das 17h34min às 18h00min, para a topologia FatTree20 foi de 221,6 pacotes por segundo. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento efetivo.

Figura 34 – Taxa média de pacotes entre os *hosts* na topologia FatTree10 (custos iniciais)



Fonte: Elaboração da própria autora.

Figura 35 – Taxa média de pacotes entre os *hosts* na topologia FatTree20 (custos iniciais)



Fonte: Elaboração da própria autora.

Comparando com os resultados da primeira simulação e observando os valores esperados da subseção 7.2.1 em relação ao desempenho da rede, verificou-se que a inclusão do Módulo Economia de Energia com custos inicias nas ligações entre os *switches* não teve relevante perda no comportamento do tráfego de dados entre os *hosts* das topologias propostas (Figura 18 e 19). Os resultados são considerados satisfatórios. Alguns *hosts* apresentaram valores abaixo do esperado, devido ao congestionamento e o atraso na transmissão de dados.

7.2.4 Terceira simulação: com o módulo economia de energia e novos custos nas ligações entre os *switches*

Na terceira simulação será verificado como se comporta o controlador Floodlight em execução com o Módulo Economia de Energia modificando os valores dos custos nas ligações entre os *switches*. Estes valores serão aplicados em cada trecho de largura de banda apresentados nas Tabelas 5 (FatTree10) e 6 (FatTree20).

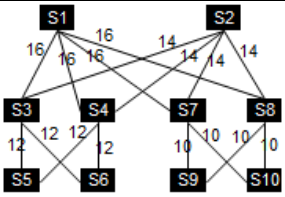
7.2.4.1 Investigação do cálculo da árvore de extensão mínima pelo algoritmo MiNet

Ainda com a segunda simulação em execução, como exemplo, foi realizada a investigação na topologia FatTree10 em relação a adaptação do algoritmo MiNet para recalcular a Árvore de Extensão Mínima até encontrar novamente uma solução para a conectividade de

todos os *switches*, sem a formação de ciclos que fecham seus vértices, e assim, conservando energia desativando algumas portas ociosas.

O grafo original da rede é mostrado na Tabela 19, os números próximos das ligações significam o seu custo. A escolha do *switch* pode ser feita randomicamente ou pelo administrador de rede.

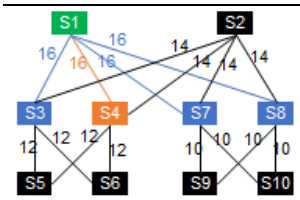
Tabela 19 – Grafo original da rede

Grafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{}	{}	{}	{s1, s2, s3, s4 s5, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

O *switch* s1 foi escolhido como ponto inicial do algoritmo. Os *switches* s3, s4, s7 e s8 estão conectados com s1 por meio de uma única ligação. Esses *switches* apresentam o mesmo custo (16) em suas ligações, então, escolhe-se qualquer ligação. O *switch* s4 foi selecionado e, portanto, a ligação {s1s4} será indicada para formar o subgrafo, apresentado na Tabela 20.

Tabela 20 – Subgrafo da rede: ligação {s1s4} selecionada

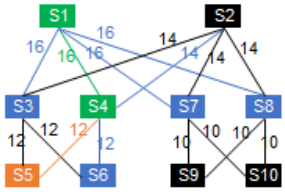
Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1}	{s1,s3}=16 {s1,s4}=16 {s1,s7}=16 {s1,s8}=16	{s1s4}	{s2, s3, s4, s5, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

O próximo *switch* escolhido é o mais perto de s1 ou s4. S3, s7 e s8 estão a uma distância 16 de s1. S2 está numa distância 14 de s4. S5 e s6 estão a uma distância 12 de s4. Os *switches*

s5 e s6 são os mais próximos (menor custo), elege-se qualquer uma dessas ligações. Logo, será escolhida a ligação {s4s5}, mostrada na Tabela 21.

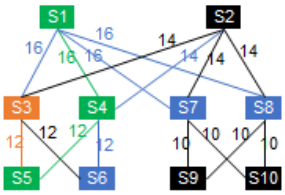
Tabela 21 – Subgrafo da rede: ligação {s4s5} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s4}	{s1,s3}=16 {s1,s7}=16 {s1,s8}=16 {s4,s2}=14 {s4,s5}=12 {s4,s6}=12	{s1s4, s4s5}	{s2, s3, s5, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

Agora, deve-se escolher o *switch* mais próximo de s1, s4 ou s5. S3, s7 e s8 estão a uma distância 16 de s1. S2 está numa distância 14 e s6 numa distância 12 de s4. S3 está a uma distância 12 de s5. Os *switches* s3 e s6 são os mais próximos (menor custo) respectivamente de s5 e s4; seleciona-se qualquer uma dessas ligações. A seguir, será escolhida a ligação {s5s3}, mostrada na Tabela 22.

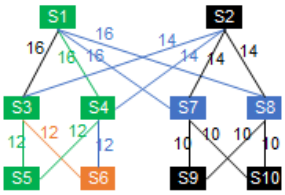
Tabela 22 – Subgrafo da rede: ligação {s5s3} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s4, s5}	{s1,s3}=16 {s1,s7}=16 {s1,s8}=16 {s4,s2}=14 {s4,s6}=12 {s5,s3}=12	{s1s4, s4s5, s5s3}	{s2, s3, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

A seguinte escolha está entre os *switches* s1, s4 e s3. S3, s7 e s8 estão a uma distância 16 de s1. S2 está numa distância 14 de s4 e s3. S6 está numa distância 12 de s3 e s4. A ligação {s1s3} está fora do subgrafo, devido à formação de ciclo. O *switch* s6 é o mais próximo (menor custo) de s3 e s4; seleciona-se qualquer uma dessas ligações. Então, será designada a ligação {s3s6}, mostrada na Tabela 23.

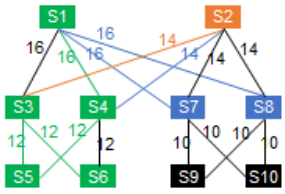
Tabela 23 – Subgrafo da rede: ligação {s3s6} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5}	{s1,s3}=16 ciclo {s1,s7}=16 {s1,s8}=16 {s4,s2}=14 {s4,s6}=12 {s3,s2}=14 {s3,s6}=12	{s1s4, s4s5, s5s3, s3,s6}	{s2, s6, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

Neste momento, os *switches* s1, s3 e s4 estão disponíveis. A ligação de {s4s6} está fora do subgrafo, devido à formação de ciclo. S7 e s8 estão a uma distância 16 de s1. S2 está numa distância 14 de s3 e s4. O *switch* s2 é o mais próximo (menor custo) de s3 e s4; elege-se qualquer uma dessas ligações. Tal posto, a ligação {s3s2} será nomeada e apresentada na Tabela 24.

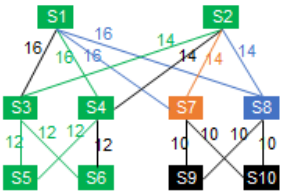
Tabela 24 – Subgrafo da rede: ligação {s3s2} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s3, s4, s5, s6}	{s1,s3}=16 ciclo {s1,s7}=16 {s1,s8}=16 {s4,s2}=14 {s4,s6}=12 ciclo {s3,s2}=14	{s1s4, s4s5, s5s3, s3,s6, s3s2}	{s2, s7, s8, s9, s10}

Fonte: Elaboração da própria autora.

Os *switches* s1 e s4 continuam no subgrafo. S2 foi adicionado. A ligação {s4s2} está fora do subgrafo, devido a formação de ciclo. S7 e s8 estão a uma distância de 16 de s1. S2 está a uma distância 14 de s7 e s8. Os *switches* s7 e s8 são os mais próximos (menor custo) de s2; seleciona-se qualquer uma dessas ligações. Sendo assim, será escolhida {s2s7}, mostrada na Tabela 25.

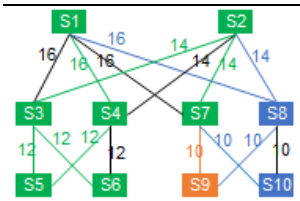
Tabela 25 – Subgrafo da rede: ligação {s2s7} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s2, s3, s4, s5, s6}	{s1,s3}=16 ciclo {s1,s7}=16 {s1,s8}=16 {s4,s2}=14 ciclo {s4,s6}=12 ciclo {s2,s7}=14 {s2,s8}=14	{s1s4, s4s5, s5s3, s3,s6, s3s2, s2s7}	{s7, s8, s9, s10}

Fonte: Elaborado pelo autor.

A escolha está entre os *switches* s1, s2 e s7. A ligação {s1s7} está fora do subgrafo, devido à formação de ciclo. S7 está a uma distância 10 de s9 e em uma distância 10 de s10. S8 está a uma distância 16 de s1 e numa distância 14 de s2. Os *switches* s9 e s10 são os mais próximos (menor custo) de s7; escolhe-se qualquer uma dessas ligações. Deste modo, será escolhida a ligação {s7s9}, mostrada na Tabela 26.

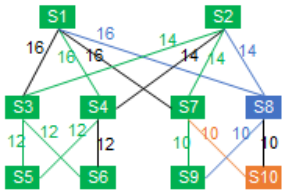
Tabela 26 – Subgrafo da rede: ligação {s7s9} selecionada

Subgrafo da rede	*Switches* dentro do subgrafo	Possíveis ligações com menor custo e sem ciclos	Ligações incluídas no subgrafo	*Switches* fora do subgrafo
	{s1, s2, s3, s4, s5, s6, s7}	{s1,s3}=16 ciclo {s1,s7}=16 ciclo {s1,s8}=16 {s4,s2}=14 ciclo {s4,s6}=12 ciclo {s2,s8}=14 {s7,s9}=10 {s7,s10}=10	{s1s4, s4s5, s5s3, s3,s6, s3s2, s2s7, s7s9}	{s8, s9, s10}

Fonte: Elaboração da própria autora.

Os *switches* s1, s2, s7 e s9 estão no subgrafo. S8 está a uma distância 16 de s1, numa distância 14 de s2 e em uma distância 10 de s9. S7 está a uma distância 10 de s10. Os *switches* s8 e s10 são os mais próximos (menor custo) respectivamente de s9 e s7; elege-se qualquer uma dessas ligações. Logo, será escolhida a ligação {s7s10}, mostrada na Tabela 27.

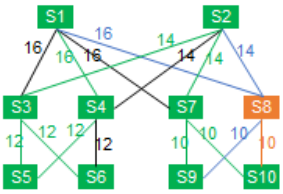
Tabela 27 – Subgrafo da rede: ligação {s7s10} selecionada

Subgrafo da rede	*Switches* dentro do subgrafo	Possíveis ligações com menor custo e sem ciclos	Ligações incluídas no subgrafo	*Switches* fora do subgrafo
	{s1, s2, s3, s4, s5, s6, s7, s9}	{s1,s3}=16 ciclo {s1,s7}=16 ciclo {s1,s8}=16 {s4,s2}=14 ciclo {s4,s6}=12 ciclo {s2,s8}=14 {s7,s10}=10 {s9,s8}=10	{s1s4, s4s5, s5s3, s3,s6, s3s2, s2s7, s7s9, s7s10}	{s8, s10}

Fonte: Elaboração da própria autora.

As opções dos *switches* s1, s2, s9 e s10 estão no subgrafo. S8 está a uma distância 16 de s1, numa distância 14 de s2, em uma distância 10 de s9 e numa distância 10 de s10. O *switch* s8 é o mais próximo (menor custo) de s9 e s10; elege-se qualquer uma dessas ligações. Portanto, é selecionada a ligação {s10s8}, mostrada na Tabela 28.

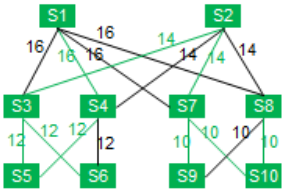
Tabela 28 – Subgrafo da rede: ligação {s10s8} selecionada

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s2, s3, s4, s5, s6, s7, s9, s10}	{s1,s3}=16 ciclo {s1,s7}=16 ciclo {s1,s8}=16 {s4,s2}=14 ciclo {s4,s6}=12 ciclo {s2,s8}=14 {s9,s8}=10 {s10,s8}=10	{s1s4, s4s5, s5s3, s3,s6, s3s2, s2s7, s7s9, s7s10, s10s8}	{s8}

Fonte: Elaboração da própria autora.

As ligações {s1s8}, {s2s8} e {s9s8} estão fora do subgrafo, devido à formação de ciclos. Aqui está o fim da investigação do subgrafo formado pelas ligações em verde que representam a Árvore de Extensão Mínima, mostrada na Tabela 18. Nesse caso, a árvore apresenta a soma de todas as suas ligações o valor 29.

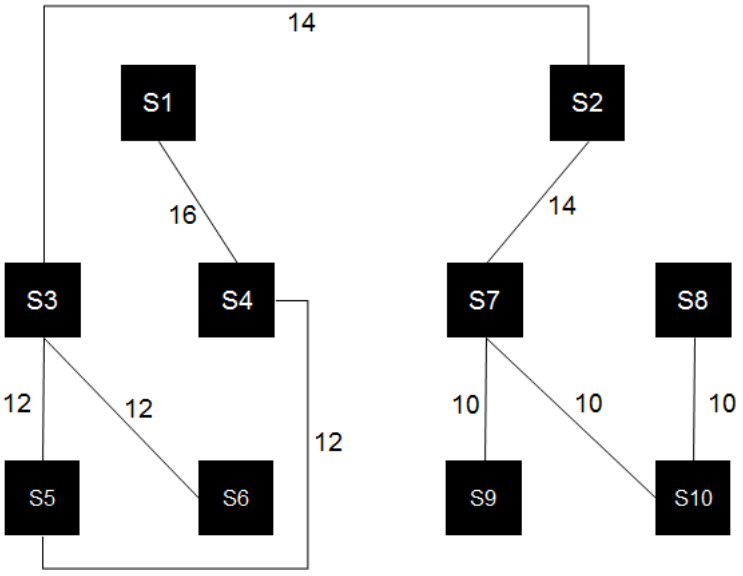
Tabela 29 – Resultado do subgrafo da rede (novos custos) com o algoritmo MiNet

Subgrafo da rede	***Switches* dentro do subgrafo**	**Possíveis ligações com menor custo e sem ciclos**	**Ligações incluídas no subgrafo**	***Switches* fora do subgrafo**
	{s1, s2, s3, s4, s5, s6, s7, s8, s9, s10}	{s1,s3}=16 ciclo {s1,s7}=16 ciclo {s1,s8}=16 ciclo {s4,s2}=14 ciclo {s4,s6}=12 ciclo {s2,s8}=14 ciclo {s9,s8}=10 ciclo	{s1s4, s4s5, s5s3, s3,s6, s3s2, s2s7, s7s9, s7s10, s10s8}	{}

Fonte: Elaboração da própria autora.

Com esta investigação, é possível evidenciar que o algoritmo MiNet identifica adequadamente a Árvore de Extensão Mínima, mantendo as portas ativas das ligações com baixo custo, exibido na Figura 36, otimizando todas as conexões de *switch* a *switch* e livre de ciclos que fecham os vértices dos *switches* na rede OpenFlow.

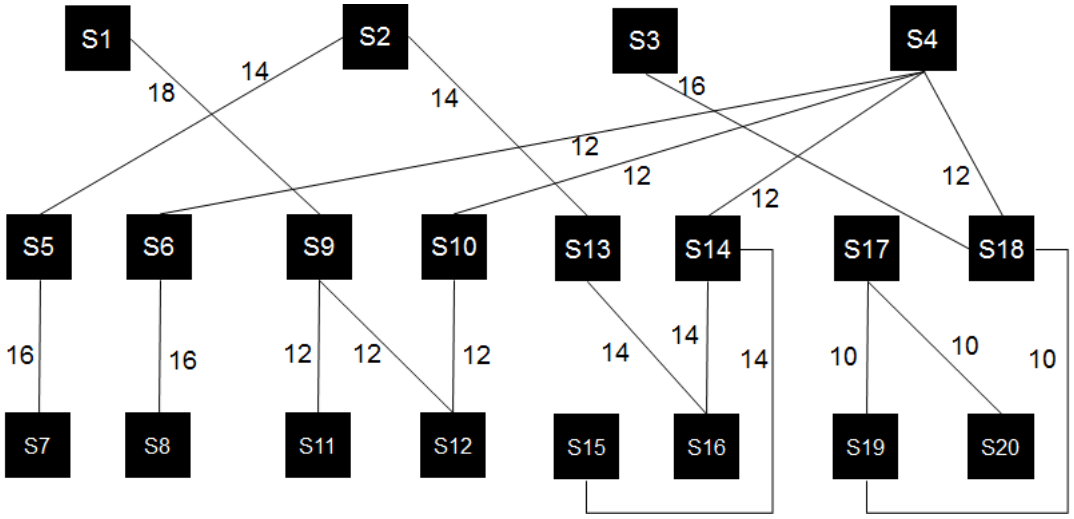
Figura 36 – Árvore de Extensão Mínima na topologia FatTree10 (novos custos)



Fonte: Elaboração da própria autora.

Do mesmo modo, realizando a investigação do cálculo da Árvore de Extensão Mínima pelo algoritmo MiNet na topologia FatTree20, será apresentada na Figura 37, o resultado do grafo da rede.

Figura 37 – Árvore de Extensão Mínima na topologia FatTree20 (novos custos)

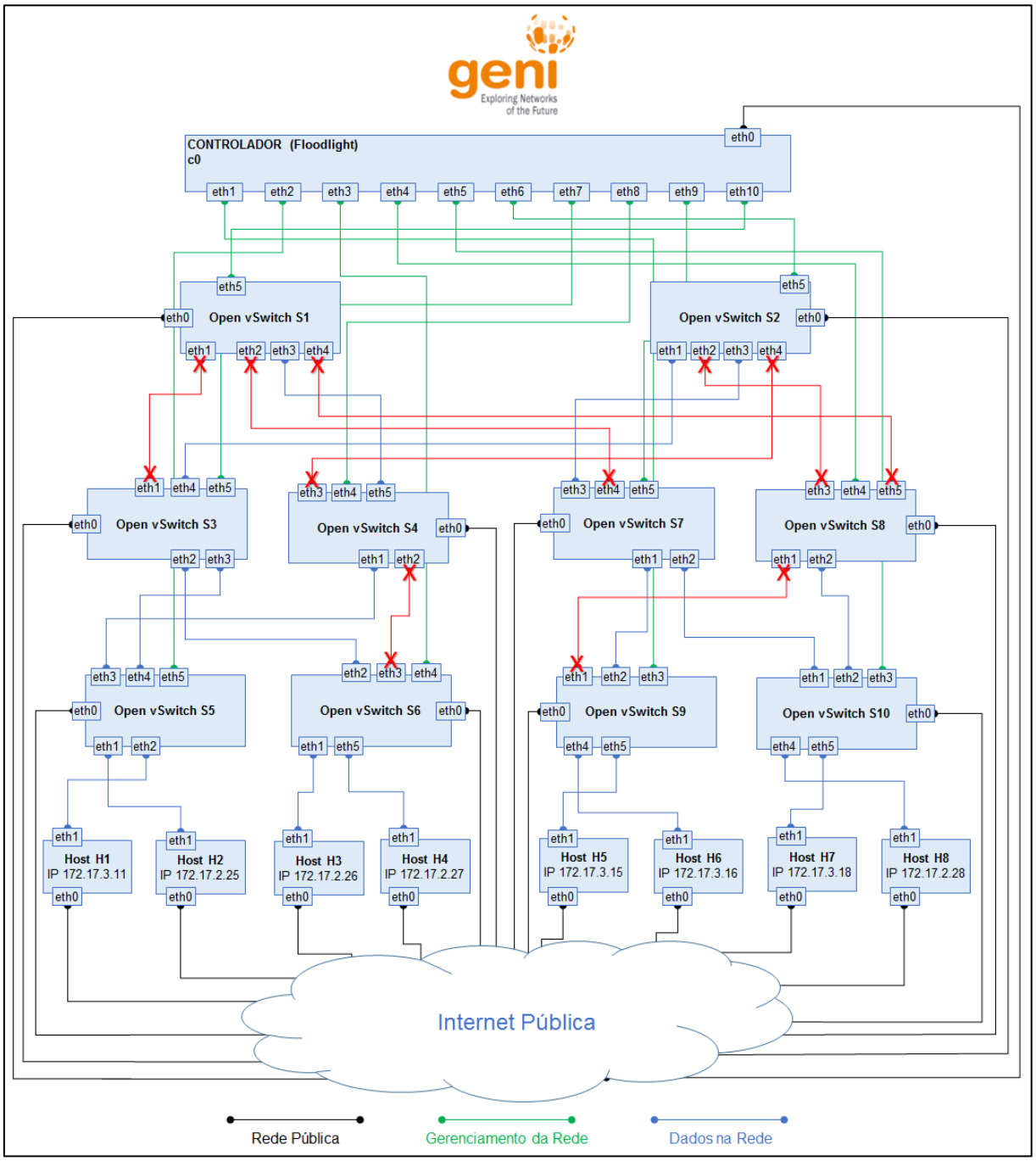


Fonte: Elaboração da própria autora.

Ressalta-se que os valores dos custos foram colocados aleatoriamente nas ligações entre os *switches*, já que a largura de banda é a mesma da segunda simulação. Nota-se, principalmente, na topologia FatTree20 (Figura 37), que as conexões com maior custo tiveram mais portas desativadas.

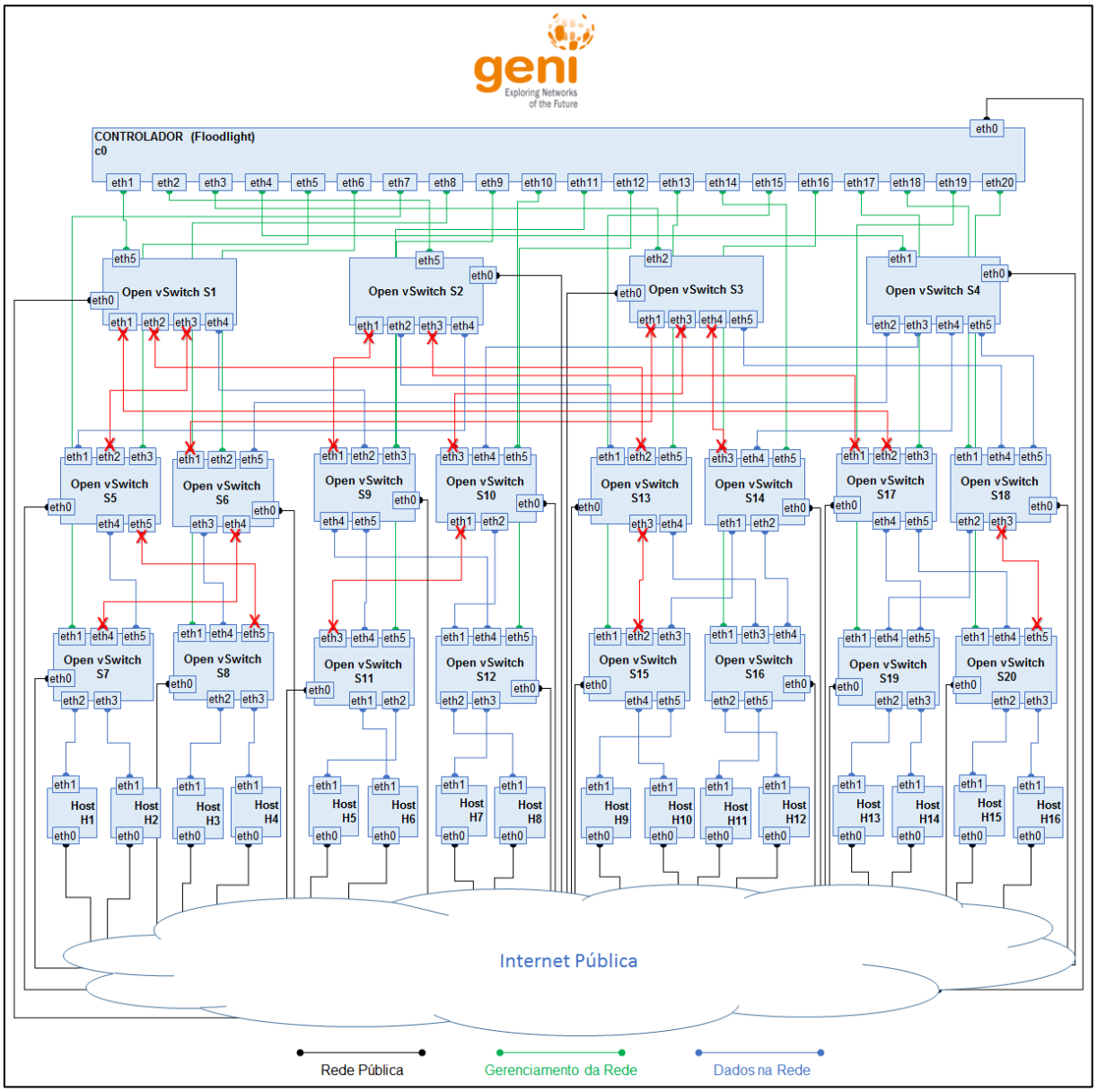
Avaliando detalhadamente com o comando “ovs-ofctl show”, pôde-se observar quais portas ficaram ativadas e quais foram desativadas nas topologias propostas exibidas nas Figuras 38 e 39.

Figura 38 – Topologia FatTree10 otimizada com o Módulo Economia de Energia (novos custos)



Fonte: Elaboração da própria autora.

Figura 39 – Topologia FatTree20 otimizada com o Módulo Economia de Energia (novos custos)



Fonte: Elaboração da própria autora.

O Quadro 12 mostra o resultado desta análise no *switch* s2 da topologia FatTree20 (Figura 39), como exemplo. A porta s2-eth1 está conectada com s9-eth1, em “config” e “state” da porta mostram o PORT_DOWN e LINK_DOWN, significa, respectivamente que a porta e a ligação estão desativadas. A porta s2-eth2 está conectada com s13-eth1, em configuração e estado da porta exibem o valor 0, significa que estão ativadas. A porta s2-eth3 está conectada com s17-eth1, em configuração e estado da porta apresentam desativada. A porta s2-eth4 está

conectada com s5-eth1, em configuração e estado da porta exibem que estão ativadas. A porta s2-eth5 corresponde a ligação no controlador e a porta s2-eth0 está conectado com a Internet pública. As portas s2-eth0 e s2-eth5 estão fora do cálculo da Árvore de Extensão Mínima.

Quadro 12 – Resultado das configurações e estado das portas do *switch* s2 da topologia FatTree20 (novos custos)

```
ligiapre@s2:~$ sudo ovs-ofctl show s2
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000002
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS
ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_VLAN SET_DL_SRC
SET_DL_DST SET_NW_SRC SET_NW_DST SET_NW_TOS SET_TP_SRC
SET_TP_DST ENQUEUE
1(eth1): addr:02:c9:4e:32:e9:9d
config:                 PORT_DOWN
    state:              LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
 2(eth2): addr:02:1b:b1:03:ae:54
    config:             0
    state:              0
    speed: 0 Mbps now, 0 Mbps max
 3(eth3): addr:02:b4:2f:b2:a7:a1
    config:             PORT_DOWN
    state:              LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
 4(eth4): addr:02:8b:71:16:2e:84
    config:             0
    state:              0
    speed: 0 Mbps now, 0 Mbps max
 LOCAL(s2): addr:e6:b7:33:39:f6:41
    config:             0
    state:              0
    speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Fonte: Adaptado de GENI (2016).

Considerando somente as ligações entre os *switches*, do mesmo modo que na segunda simulação, para a topologia FatTree10 do total de quarenta portas Ethernet (100%), a

sobreposição da Árvore de Extensão Mínima desativaria quatorze portas Ethernet dos *switches* (redução de 35%), sendo, s1-eth1, s1-eth2, s1-eth4, s2-eth2, s2-eth4, s3-eth1, s4-eth2, s4-eth3, s6-eth3, s7-eth4, s8-eth1, s8-eth3, s8-eth5, s9-eth1. O consumo de energia total das portas Ethernet seria de 65%.

Já para a topologia FatTree20, considerando somente as ligações entre os *switches*, do total de oitenta portas Ethernet (100%) a sobreposição da Árvore de Extensão Mínima desativaria vinte e seis portas Ethernet dos *switches* (redução de 32,5%), sendo, s1-eth1, s1-eth2, s1-eth3, s2-eth1, s2-eth3, s3-eth1, s3-eth3, s3-eth4, s5-eth2, s5-eth5, s6-eth1, s6-eth4, s7-eth4, s8-eth5, s9-eth1, s10-eth1, s10-eth3, s11-eth3, s13-eth2, s13-eth3, s14-eth3, s15-eth2, s17-eth1, s17-eth2, s18-eth3, s20-eth5. O consumo de energia total das portas Ethernet seria de 67,5%.

7.2.4.2 Avaliação do consumo de energia durante a execução da topologia proposta

Verificando-se novamente os registros com os testes em andamento da segunda simulação da topologia FatTree10, às 09h30min03s283ms foram alterados os custos das ligações entre os *switches*. Nota-se à ativação de algumas portas Ethernet que estavam desativadas e o desligamento de outras portas Ethernet que estavam ativas, devido à atribuição de novos custos entre as conexões dos *switches*. A partir deste período foi inserido tráfego na rede e não observou mudança na topologia em relação as portas desativadas (Figura 38). A simulação encerrou-se às 10h00min17s487ms. Portanto, o consumo de energia final das portas Ethernet para este período foi de 65% (economia de 35%), considerado satisfatório para economia de energia na rede. Notoriamente esta porcentagem poderia variar se a simulação continuasse por mais tempo, de acordo com o fluxo de dados na rede e a mudança de novos custos nas ligações entre os *switches*.

Analisando precisamente os registros com os testes em andamento da segunda simulação da topologia FatTree20, às 18h00min25s283ms foram alterados os custos das ligações entre os *switches*, do mesmo modo, que na topologia anterior, verifica-se o similar comportamento em ativar algumas portas Ethernet que estavam desativadas e o desligamento de outras portas Ethernet que estavam ativas, devido à atribuição de novos custos entre as conexões dos *switches*. A partir deste período foi inserido tráfego na rede e não se observou mudança na topologia em relação às portas desativadas (Figura 39). A simulação encerrou-se às 18h30min10s115ms. Portanto, o consumo de energia final das portas Ethernet para este período foi de 67,5% (economia de 32,5%), considerado satisfatório para economia de energia

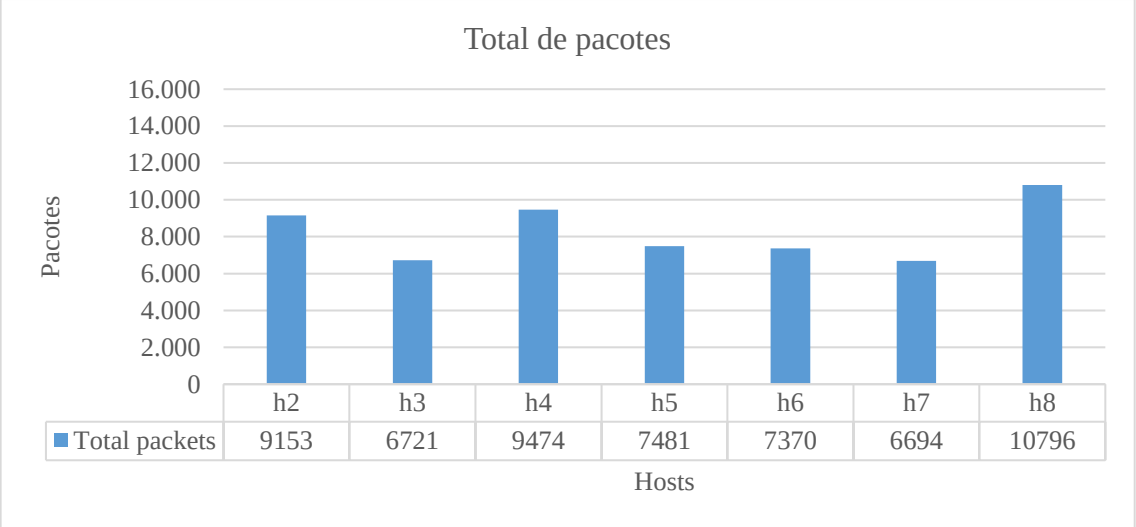
na rede. Evidentemente que esta porcentagem poderia variar se a simulação continuasse por um longo período de tempo, de acordo com o fluxo de dados na rede e a mudança de novos custos nas ligações entre os *switches*.

7.2.4.3 Análise de desempenho da rede sobre o tráfego de dados gerado pelo D-ITG

Durante o período da simulação em ambas topologias, foram utilizadas a configurações de tráfego de dados comentada na subseção 7.2.1. Em seguida, serão apresentadas as métricas empregadas (total de pacotes, bytes recebidos e taxa média de pacotes) para medir o desempenho do Módulo Economia Energia com novos custos nas ligações entre as ligações dos *switches* (Apêndice C). O fluxo de dados é transmitido pelos *hosts* ao receptor h1 e vice-versa nas duas topologias.

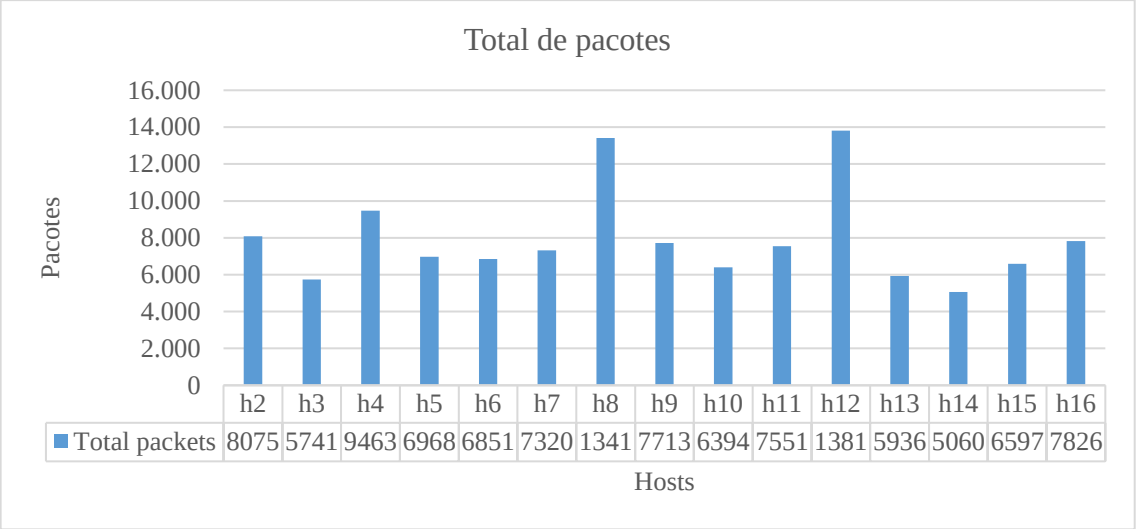
É conferido o total de pacotes em cada topologia apresentados nas Figuras 40 e 41. Observa-se que no período das 09h30min às 10h00min, a média de pacotes para a topologia FatTree10 foi de 8.241 pacotes e no período das 18h00min às 18h30min para a topologia FatTree20 foi de 7.915 pacotes. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego o controlador teve um comportamento apropriado.

Figura 40 – Total de pacotes entre os *hosts* na topologia FatTree10 (novos custos)



Fonte: Elaboração da própria autora.

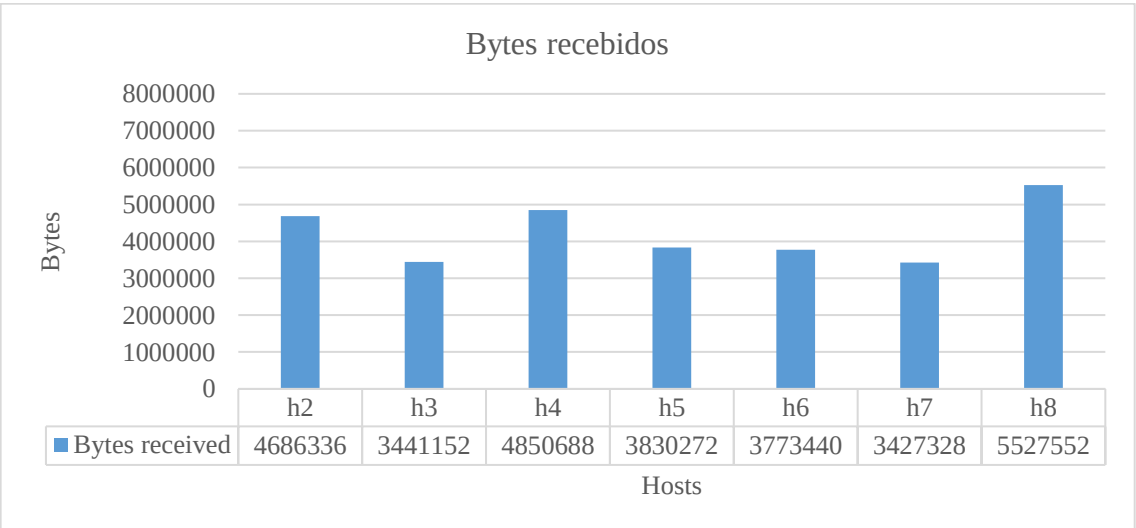
Figura 41 – Total de pacotes entre os *hosts* na topologia FatTree20 (novos custos)



Fonte: Elaboração da própria autora.

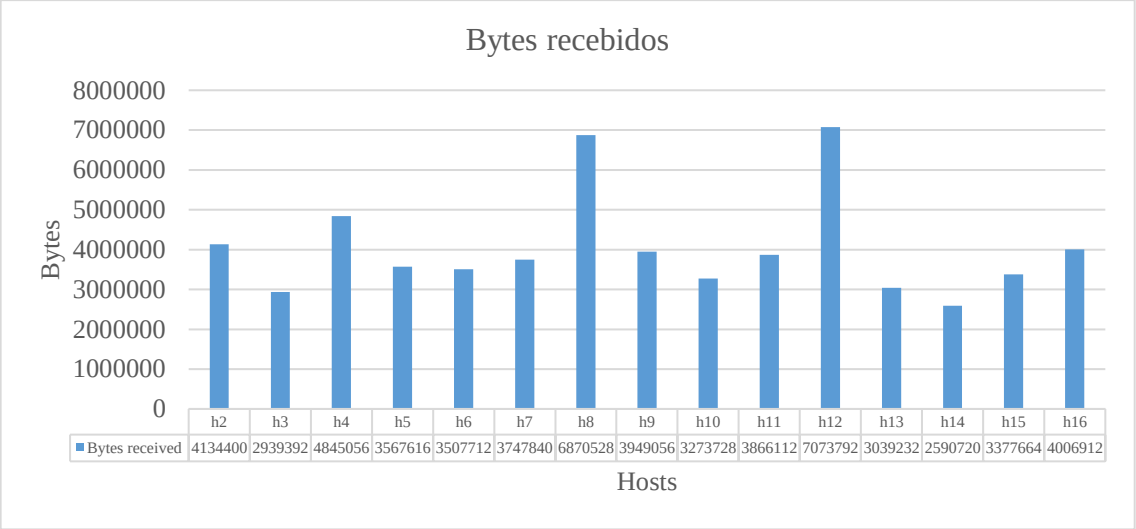
Do mesmo modo, nas Figuras 42 e 43, são apurados a quantidade de bytes recebidos em cada topologia. Observa-se que no período das 09h30min às 10h00min a média de bytes convertido em megabytes para a topologia FatTree10 foi de 4,21 MB e no período das 18h00min às 18h30min para a topologia FatTree20 foi de 4,05 MB. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento pertinente.

Figura 42 – Bytes recebidos entre os *hosts* na topologia FatTree10 (novos custos)



Fonte: Elaboração da própria autora.

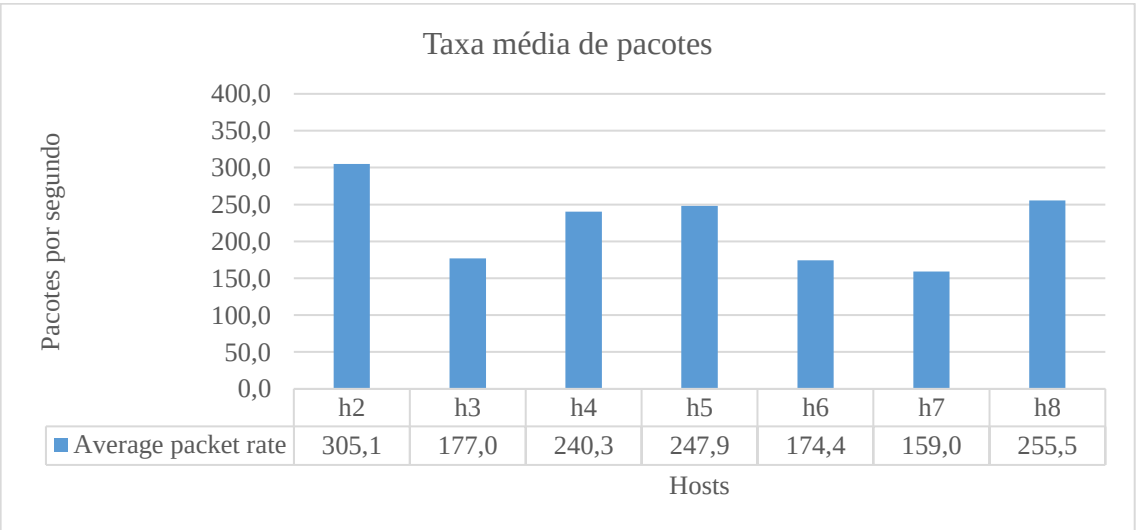
Figura 43 – Bytes recebidos entre os *hosts* na topologia FatTree20 (novos custos)



Fonte: Elaboração da própria autora.

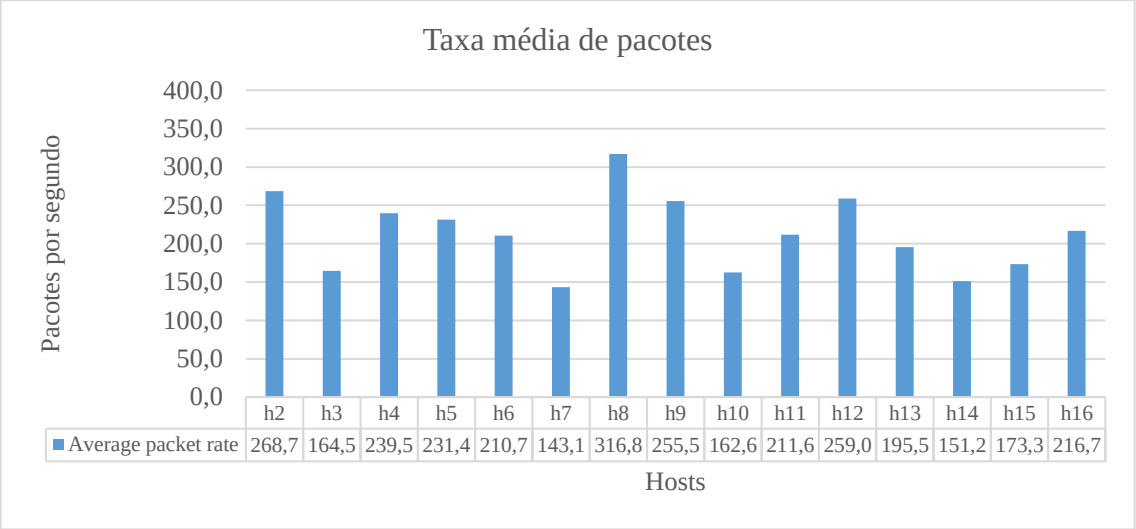
É examinada a taxa média de pacotes por segundo mostradas nas Figuras 44 e 45 em cada topologia. Nota-se que no período das 09h30min às 10h00min a média da taxa de pacotes foi de 222,7 pacotes por segundo para a topologia FatTree10 e, no período das 18h00min às 18h30min, para a topologia FatTree20, foi de 213,3 pacotes por segundo. Comparando o escalonamento da rede de 10 para 20 *switches* com a mesma configuração de tráfego, o controlador teve um comportamento efetivo.

Figura 44 – Taxa média de pacotes entre os *hosts* na topologia FatTree10 (novos custos)



Fonte: Elaboração da própria autora.

Figura 45 – Taxa média de pacotes entre os *hosts* na topologia FatTree20 (novos custos)



Fonte: Elaboração da própria autora.

Analisando com os resultados da primeira simulação e observando os valores esperados da subseção 7.2.1 em relação ao desempenho da rede, verificou-se que a inclusão do Módulo Economia de Energia com novos custos nas ligações entre os *switches* não teve relevantes perdas no comportamento do tráfego de dados entre os *hosts* das topologias propostas (Figura 18 e 19). Os resultados são considerados satisfatórios. Alguns *hosts* apresentaram valores abaixo do esperado, devido ao congestionamento e o atraso na transmissão de dados.

7.2.5 Comparação e análise de desempenho da rede sobre o tráfego de dados gerado pelo D-ITG nas três simulações

No decurso da simulação foram analisados outros parâmetros para verificação do desempenho da topologia Fat Tree na rede OpenFlow com o controlador Floodlight sem/com o Módulo Economia de Energia no controlador Floodlight. Tais métricas se referem a vazão, média de atraso e variação estatística de atraso. Do mesmo modo, foram utilizadas as configurações do tráfego de dados comentada na subseção 7.2.1.

A comparação será com as três simulações simultaneamente de cada *host* das topologias FatTree10 e FatTree 20, analisado nas seguintes fases:

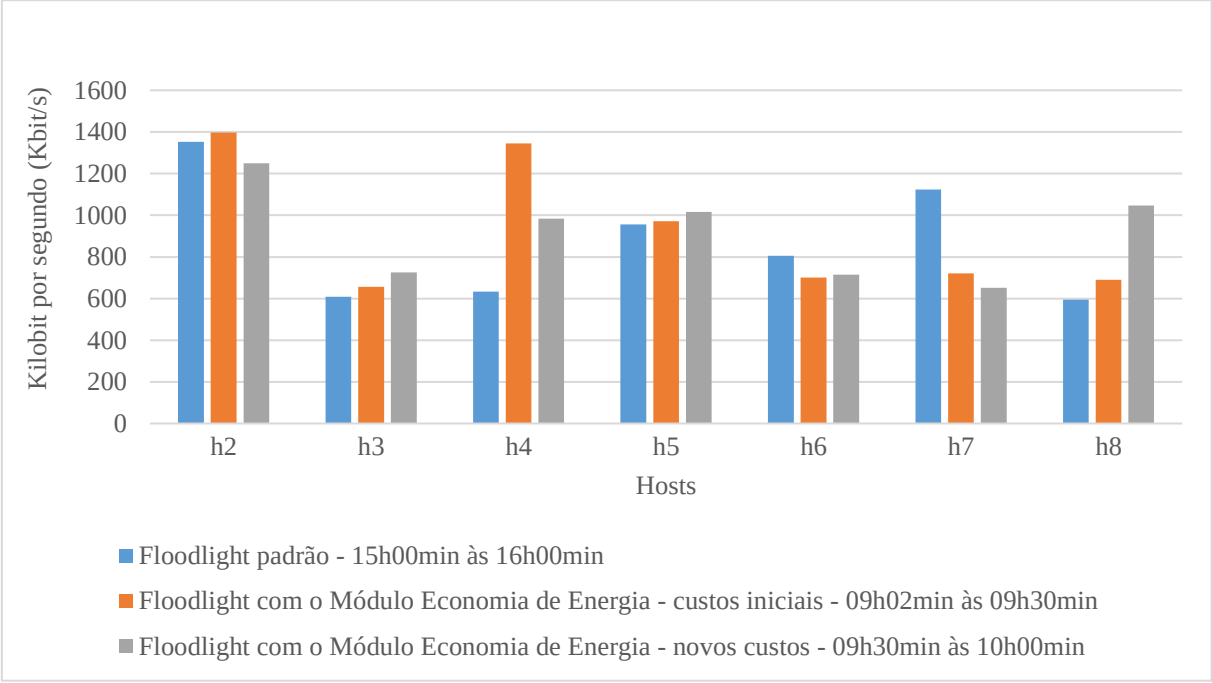
a) No modo padrão do controlador Floodlight (sem o Módulo Economia de Energia) nos períodos: 15h00min às 16h00min (topologia FatTree10) e das 19h00min às 20h00min (topologia FatTree20);

b) Com o Módulo Economia de Energia e os custos iniciais nas ligações entre os *switches* nos períodos: 09h02min às 09h30min (topologia FatTree10) e das 17h34min às 18h00min (topologia FatTree20);

c) Com o Módulo Economia de Energia e novos custos nas ligações entre os *switches* nos períodos: 09h30min às 10h00min (topologia FatTree10) e das 18h00min às 18h30min (topologia FatTree20).

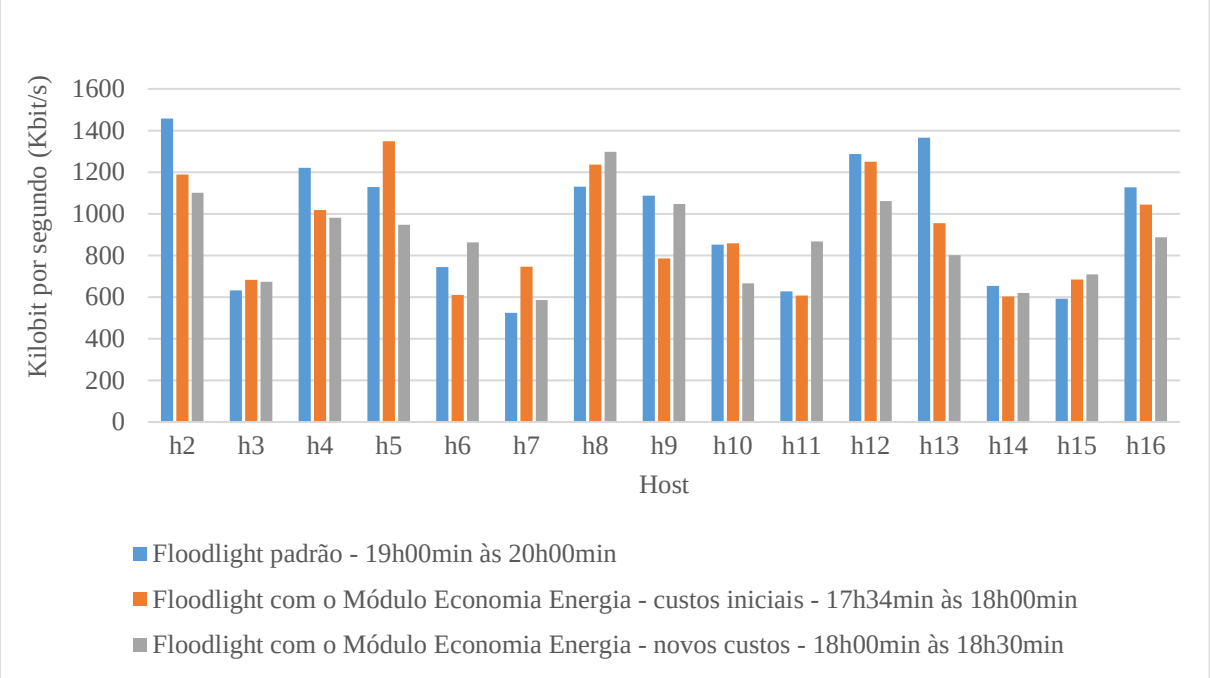
Observa-se nas Figuras 46 (FatTree10) e 47 (FatTree20) que em relação a vazão (taxa de transferência de bits) os valores transmitidos pelos *hosts* nas fases das simulações ficaram próximos de 1.000 kilobits por segundo, devido à largura de banda estipulada em 2 Mbps para os enlaces que interligam os *hosts* aos *switches*. Averiguando os números esperados da subseção 7.2.1 em relação ao desempenho da rede, verificou-se que a inclusão do Módulo Economia de Energia (custos iniciais e novos custos) nas ligações entre os *switches* não teve relevante perda no comportamento do tráfego de dados entre os *hosts*. Os resultados são considerados satisfatórios. Alguns *hosts* apresentaram valores abaixo do esperado, devido ao congestionamento e o atraso na transmissão de dados.

Figura 46 – Comparação da vazão de cada *host* para a topologia FatTree10



Fonte: Elaboração da própria autora.

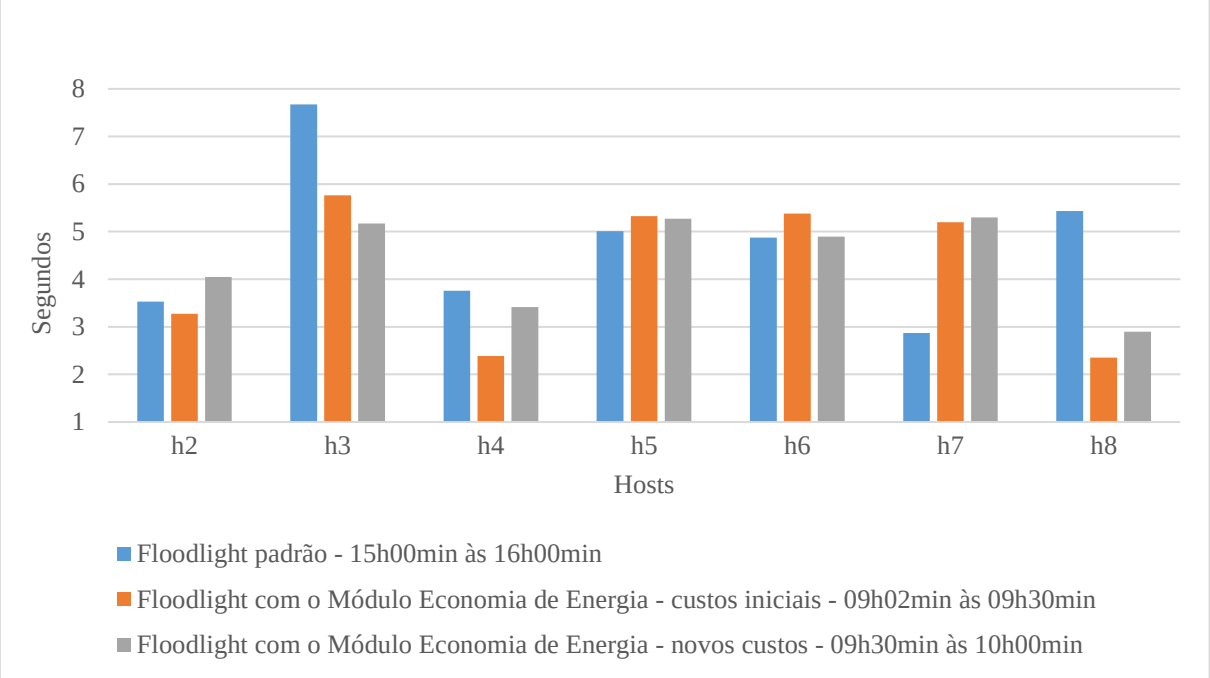
Figura 47 – Comparação da vazão de cada *host* para a topologia FatTree20



Fonte: Elaboração da própria autora.

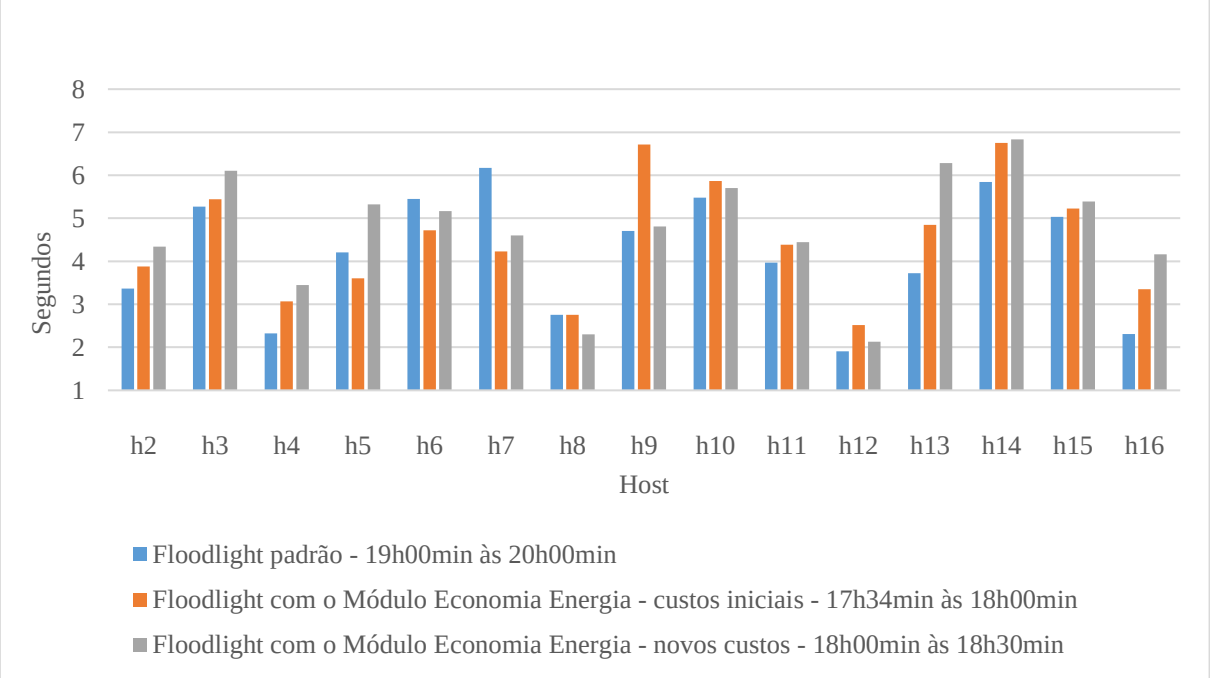
Verifica-se nas Figuras 48 (FatTree10) e 49 (FatTree20) que em relação à média de atraso, ou seja, o tempo que demora um bit, desde que parte de um emissor até o seu destinatário, teve oscilações para todas as fases, mas sem relevância considerável, uma vez que o tempo não foi superior a oito segundos e, além disso, têm-se inúmeros projetos alocados na própria infraestrutura do GENI usando outros recursos ao mesmo tempo provocando o atraso. Em destaque, observa-se que nas simulações com a inclusão de custos iniciais e novos custos teve alguns períodos de atraso mais longos. Foram nesses momentos que algumas portas Ethernet dos *switches* ficaram desativadas.

Figura 48 – Comparação da média de atraso de cada *host* para a topologia FatTree10



Fonte: Elaboração da própria autora.

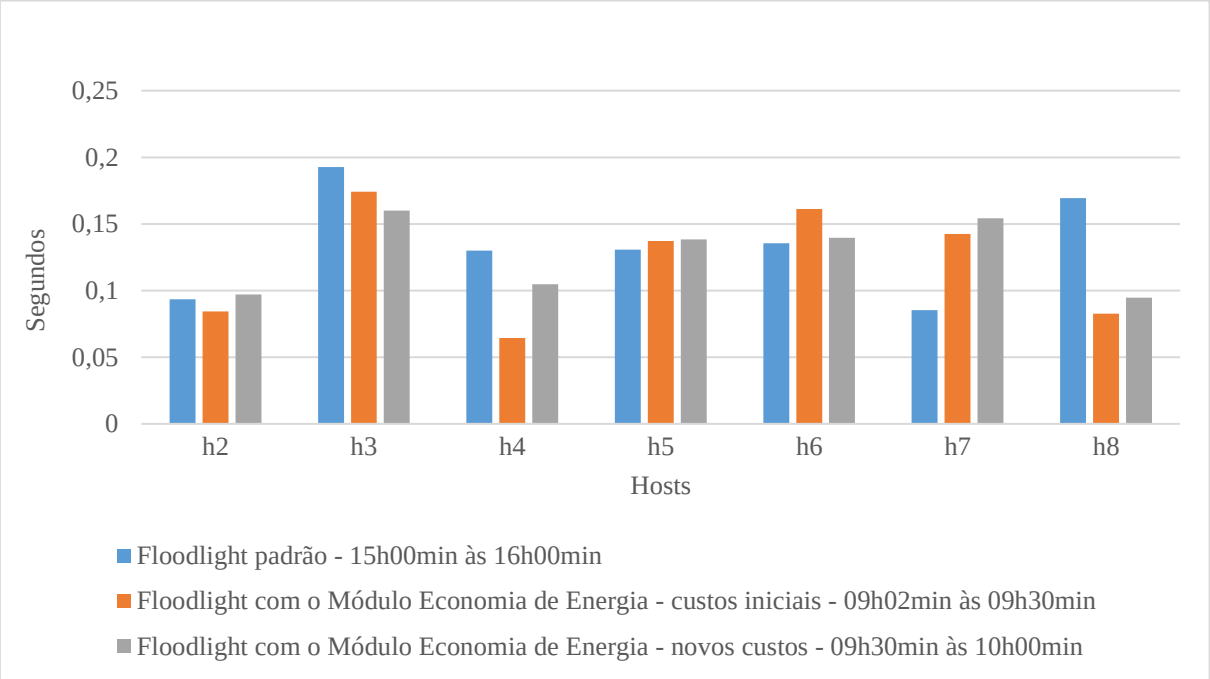
Figura 49 – Comparação da média de atraso de cada *host* para a topologia FatTree20



Fonte: Elaboração da própria autora.

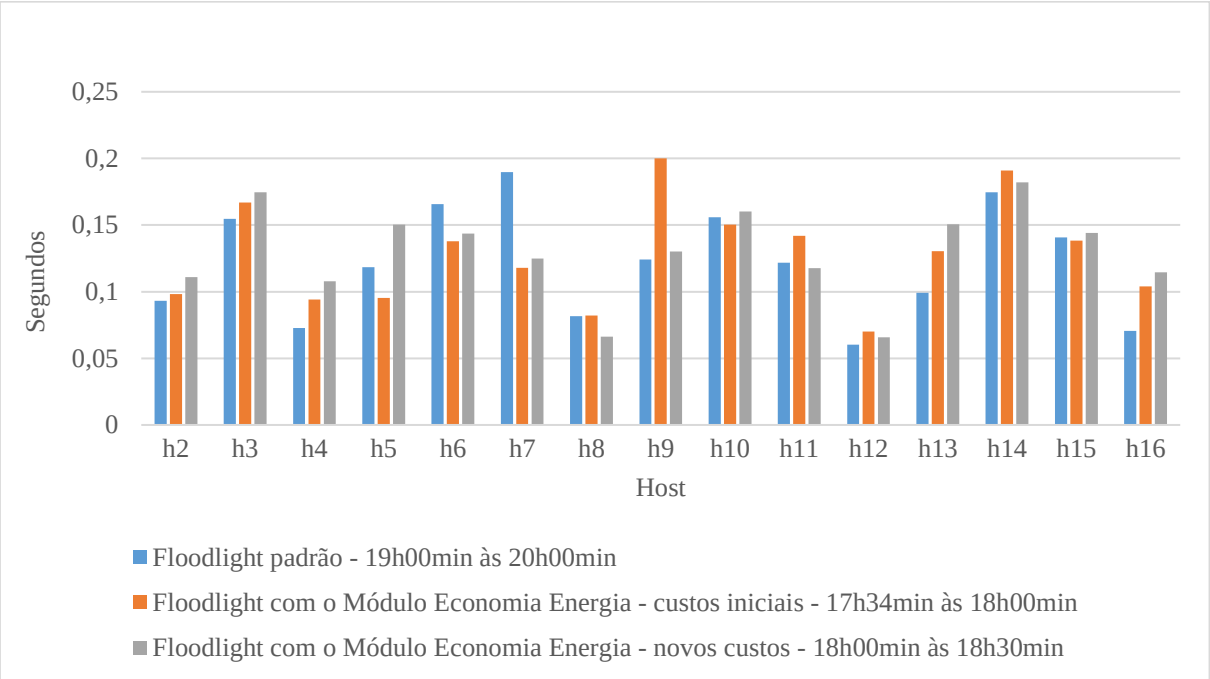
E, por fim, nas Figuras 50 (FatTree10) e 51 (FatTree20), analisa-se a variação estatística de atraso na entrega de dados em uma rede. Observa-se que uma variação de atraso elevada produz uma recepção não regular dos pacotes, o que não se ressalta nas fases simuladas. Averígua-se valores próximos a zero para todos os nós.

Figura 50 – Comparação da variação estatística de atraso de cada *host* para a topologia FatTree10



Fonte: Elaboração da própria autora.

Figura 51 – Comparação da variação estatística de atraso de cada *host* para a topologia FatTree20



Fonte: Elaboração da própria autora.

8 DISCUSSÕES FINAIS

A ideia de desativar a infraestrutura de rede tem sido considerada um tabu. Qualquer sistema de gerenciamento de energia dinâmica que aposta alcançar o equilíbrio de energia desligando um subconjunto de componentes ociosos deve demonstrar que os componentes ativos podem atender à carga atual oferecida, bem como a posterior mudança de carga. A economia de energia deve ser conveniente, os efeitos no desempenho devem ser mínimos, e a tolerância a falhas não devem ser sacrificados. O sistema deve produzir um conjunto viável de subconjuntos de rede que podem encaminhar dados para todos os *hosts*, e ser capaz de escalar para milhares de servidores.

Neste estudo, foi desenvolvido o Módulo Economia de Energia no controlador Floodlight e testado na plataforma experimental de Redes Definidas por Software GENI. Este ambiente é uma federação que possui um robusto laboratório para experimentações em redes de computadores reais ou virtuais em larga escala. Analisando os resultados nas duas topologias criadas sobre os recursos de GENI, pôde-se observar que ao escalonar a rede, não produziu expressiva perda de desempenho no tráfego de dados com o módulo incluído no controlador, comparado nas três simulações. Após a inicialização do controlador Floodlight com o Módulo Economia de Energia, as portas Ethernet dos *switches* que foram desativadas, mantiveram-se assim, até a inserção de novos custos entre as ligações dos *switches*, com exceção na topologia FatTree20 que apresentou, na fase de inclusão de custos iniciais, a ativação de duas portas Ethernet durante a simulação, devido ao fluxo de dados na rede. Ressalta-se que os valores dos custos foram sugeridos conforme a largura de banda de cada conexão, que neste caso foram iguais para cada camada da rede FatTree. Logicamente poderia ser induzida nas topologias criadas, largura de banda mais elevada para determinadas ligações e alocar valores de custos mais baixos para forçar a criação da Árvore de Extensão Mínima nestes enlaces com maior largura de banda, evitando o congestionamento na rede, e assim, as portas Ethernet com menor banda seriam desativadas, sem comprometer o desempenho da rede.

O consumo de energia final concluído neste trabalho, considerando somente as ligações entre os *switches*, sendo, quarenta portas Ethernet para a topologia FatTree10 e oitenta portas Ethernet para a topologia FatTree20, logo após os aproximados sessenta minutos de simulação, foi respectivamente de 65% (economia de 35% para a topologia FatTree10) e 67,5% (economia de 32,5% para a topologia FatTree20) considerados satisfatórios para a eficiência energética em redes de centros de dados, de acordo com a literatura pesquisada.

Por fim, as três simulações descritas neste trabalho mostram que a implementação proposta pelo algoritmo MiNet para cálculo/recálculo da Árvore de Extensão Mínima no Módulo Economia de Energia implantado no controlador Floodlight, com o objetivo de economizar energia na rede, desativando portas ociosas, comportam-se corretamente com resultados adequados para eficiência energética na rede. A validação é considerada positiva para a extensão de redes mais complexas, como topologias Fat Tree (árvore gorda). A proposta de inovação desses algoritmos foi devida às novas tendências em Internet do Futuro, na qual a tomada de decisão passa a ser centralizada em controladores e, principalmente, pelo fato do algoritmo possuir código fonte aberto, no qual os administradores de rede podem fazer edições no código fonte, aprimorando a sua eficácia. Os algoritmos existentes apresentam código fonte fechado pelos fabricantes dos *switches* tradicionais e, desse modo, os administradores de rede não podem inovar configurações para melhorar as condições da rede.

9 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Neste trabalho foi realizado um estudo sobre Internet do Futuro (IF), apresentando o andamento da pesquisa experimental no Brasil e no mundo, e mostrando que de uma maneira rápida e simplificada é possível construir infraestrutura para experimento de protocolos baseadas em redes OpenFlow. Observa-se que é possível iniciar análise para IF virtualizando todo plano de dados alinhado às necessidades de um ambiente real. O paradigma de SDN, implementado em OpenFlow, permite projetar rapidamente e validar algoritmos inovadores para o controle da rede.

Atualmente, os ambientes de TIC “verde” estão se tornando uma realidade. Novos serviços e produtos serão ofertados, tendo como foco uma maior eficiência energética, proporcionando o surgimento de novos negócios. Desse modo, pode-se ter um equilíbrio entre a utilização de recursos naturais e a oferta de novos serviços.

A prática de TIC, principalmente a que trata da eficiência energética, visa otimizar o uso da energia utilizada pelos equipamentos de informática. Essas, quando aplicadas, podem trazer benefícios para todos, empresas com a redução de custo, sociedade com melhoria da qualidade de vida, e o meio ambiente com sua preservação, tornando o mundo mais sustentável.

O trabalho explorou a viabilidade de configurar uma topologia Fat Tree utilizando tecnologias SDN. Essa topologia é típica de centro de dados densos. O módulo proposto com código fonte aberto, aborda topologias com ligações livres de ciclos entre os *switches* na camada 2, promovendo eficiência energética. O Módulo Economia de Energia oferece essa funcionalidade, evitando as desvantagens das soluções clássicas não OpenFlow, como STP, permitindo que os aplicativos especifiquem dinamicamente a métrica que o controlador usa para a construção da Árvore de Extensão Mínima.

O algoritmo MiNet implantado no Módulo Economia de Energia, ambos com código fonte aberto, permitem a redução do consumo de energia, desligando as interfaces inativas e possibilitam alterações pelos operadores de redes conforme as suas necessidades de configurações. Validações iniciais mostram que o módulo se comporta como esperado, resolvendo os problemas causados por ciclos entre dispositivos Ethernet e reagindo corretamente em caso de mudanças na topologia de redes pequenas e mais amplas.

Considerando o sucesso do paradigma SDN e todos os desafios identificados neste estudo, observa-se que existe um vasto campo para o desenvolvimento de novos trabalhos de pesquisa focado em SDN.

Trabalhos futuros sobre Árvore de Extensão Mínima poderão estar focados na elaboração e adaptação de outros algoritmos para comparar a eficiência do MiNet em economizar energia nas redes. Novos recursos poderão ser incluídos, entre eles, a definição de regras em interfaces para bloquear a transmissão de dados por um período limitado de tempo antes de ligá-las. O desempenho do módulo poderá ser avaliado sobre outras topologias mais complexas a fim de medir os tempos de reação e descobrir qualquer limitação do algoritmo.

Os próximos estudos consistirão em atribuir uma maior quantidade de tráfego de dados entre os *hosts,* por um período mais longo de tempo e avaliar o desempenho do módulo em ativar e desativar as portas Ethernet para economizar energia nos *switches* OpenFlow. Outra etapa a ser almejada, será solicitar *hosts* físicos na federação GENI para testar o módulo em um ambiente de rede física e, assim, concretizar a eficácia do Modulo Economia de Energia. E assim, obtendo novos resultados, estes poderão ser submetidos a publicação em periódicos científicos da área.

9.1 PUBLICAÇÕES

DE OLIVEIRA, ROGERIO LEAO SANTOS; SCHWEITZER, CHRISTIANE MARIE; SHINODA, AILTON AKIRA; LIGIA RODRIGUES PRETE. Using Mininet for emulation and prototyping Software-Defined Networks. In: 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, Bogota, Colombia. **Proceedings...**Colombia: IEEE Colombian Conference on Communications and Computing (COLCOM). p. 1.

PRETE, LIGIA RODRIGUES; SHINODA, AILTON AKIRA; SCHWEITZER, CHRISTIANE MARIE; DE OLIVEIRA, ROGERIO LEAO SANTOS. Simulation in an SDN network scenario using the POX Controller. In: 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, Bogota, Colombia. **Proceedings...**Colombia: IEEE Colombian Conference on Communications and Computing (COLCOM). p. 1.

OLIVEIRA, R. L. S.; SHINODA, A. A.; SCHWEITZER, C. M.; IOPE, R. L.; PRETE, L. R. L3-ARPSec: A Secure Openflow Network Controller Module to control and protect the Address Resolution Protocol. In: XXXIII Simpósio Brasileiro de Telecomunicações (SBrT 2015), 2015, Juiz de Fora. **Anais...** Juiz de Fora, 2015. v. 33. p. 157-161.

PRETE, L. R.; SCHWEITZER, C. M.; SHINODA, A. A.; OLIVEIRA, R. L. S. Árvore de Extensão Mínima no Processo de Conservação de Energia em Redes Definidas por Software. In: X WORKSHOP DE PÓS-GRADUAÇÃO E PESQUISA DO CENTRO PAULA SOUZA, 2015, São Paulo. **Anais...** São Paulo, 2015. v. 10.

PRETE, L. R.; SCHWEITZER, C. M.; SHINODA, A. A.; OLIVEIRA, R. L. S. Experimentação em Redes Definidas por Software empregando a plataforma GENI. In: XI WORKSHOP DE PÓS-GRADUAÇÃO E PESQUISA DO CENTRO PAULA SOUZA, 2016, São Paulo. **Anais...** São Paulo, 2016. v. 11.

OLIVEIRA, R. L. S.; SCHWEITZER, C. M.; SHINODA, A. A.; PRETE, L. R. Testes de desempenho com o módulo de segurança L3-ARPSec em Redes Definidas por Software. In: XI WORKSHOP DE PÓS-GRADUAÇÃO E PESQUISA DO CENTRO PAULA SOUZA, 2016, São Paulo. **Anais...** São Paulo, 2016. v. 11.

PRETE, L. R.; SCHWEITZER, C. M.; SHINODA, A. A.; OLIVEIRA, R. L. S. **Power management on the OpenFlow network switches with Energy Saving Module implemented to the Floodlight Controller**. Computers & Electrical Engineering, 2016. Em submissão.

REFERÊNCIAS

ALVES, V. H. C. **Estratégias de TI verde podem ajudar a reduzir emissões de CO_2**. São Paulo: TI Inside Online, 2010. Disponível em: <http://www.tiinside.com.br/30/06/2010/estrategias-de-ti-verde-podem-ajudar-a-reduzir-emissoes-de-co-/ti/189076/news.aspx>. Acesso em: 14 ago. 2013.

BALAKRISHNAN, H.; LAKSHMINARAYANAN, K.; RATNASAMY, S.; SHENKER, S.; STOICA, I.; WALFISH, M. **A layered naming architecture for the internet**. **Comput. Commun. Rev**., New York, v. 34, n. 4, p. 343–352, 2004.

BHATIA, S.; MOTIWALA, M.; MUHLBAUER, W.; MUNDADA, Y.; VALANCIUS, V.; BAVIER, A.; FEAMSTER, N.; PETERSON, L.; REXFORD, J. Trellis: a platform for building flexible, fast virtual networks on commodity hardware. In: CONEXT CONFERENCE, CoNEXT, 8., 2008, New York. **Proceedings...**New York: ACM, 2008, v.72, p. 1–6.

BONFIRE. **Building service testbeds on future internet research and experimentation**. Barcelona: University of Southampton IT Innovation Centre, 2013. Disponível em: <http://www.bonfire-project.eu>. Acesso em: 20 mar 2013.

CASADO, M.; KOPONEN, T.; RAMANATHAN, R.; SHENKER, S. Virtualizing the Network Forwarding Plane. In: WORKSHOP ON PROGRAMMABLE ROUTERS FOR EXTENSIBLE SERVICES OF TOMORROW, PRESTO, 10., 2010, Philadelphia. **Proceedings...** New York: ACM Press, 2010.v. 8, p. 1–8.

CHOWDHURY, N. M. K.; BOUTABA, R. **A survey of network virtualization**. **Comput. Netw.,** New York, v. 54, n. 5, p. 862–876, 2010.

COSTA, L. R. **Openflow e o paradigma de redes definidas por software.** 2013. 155 f. Trabalho de Conclusão de Curso (Graduação em Curso de Computação) – Instituto de Ciências Exatas, Universidade de Brasília, Brasília, DF, 2013.

CREW. **Cognitive radio experimentation world**. Belgium: CREW – Cognitive Radio Experimentation World, 2009. Disponível em: <http://www.crew-project.eu>. Acesso em: 22 mar 2013.

DROUANT, N; RONDEAU E.; GEORGES J. P.; LEPAGE F. Designing green network architectures using the ten commandments for a mature ecosystem. **Comput. Commun.**, Netherlands, v. 42, p. 38–46, Apr. 2014.

EGI, N.; GREENHALGH, A.; HANDLEY, M.; HOERDT, M.; HUICI, F.; MATHY, L.; PAPADIMITRIOU, P. A platform for high performance and flexible virtual routers on commodity hardware. **Comput. Commun. Rev.,** New York, v. 40, n. 1, p. 127–128, 2010.

EIDE, E.; STOLLER, L.; STACK, T.; FREIRE, J.; LEPREAU, J. Integrated scientific workflow management for the emulab network testbed. In: ANNUAL CONFERENCE ON USENIX; ANNUAL TECHNICAL CONFERENCE, 6., 2006, Berkeley. **Proceedings...** Berkeley: USENIX Association, 2006. p. 33.

EMULAB. **Network emulation testbed**. Salt Lake: The University of Utah, 2016. Disponível em: <http://www.emulab.net>. Acesso em: 29 mar 2013.

FARIAS, F. N. N.; DIASJÚNIOR, J. M.; SALVATTI, J. J.; SILVA, S.; ABELÉM, A. J. G.; SALVADOR, M. R.; STANTON, M. A. Pesquisa experimental para a internet do futuro: uma proposta utilizando virtualização e o framework openflow. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 29., 2011, Campo Grande. **Anais...** Campo Grande: [s.n], 2011. p. 1–62.

FIBRE. **Future internet brazilian environment for experimentation**. Brasil: RNP, 2010. Disponível em: <https://fibre.org.br/>. Acesso em: 10 maio 2016.

FIRE. **Future internet research and experimentation:** an overview of european fire initiative and its projects. European Union: Publications Office of the European Union, 2010. Technical Report. Disponível em: <http://cordis.europa.eu/fp7/ict/fire/docs/fire-brochure-201_en.pdf>. Acesso em: 18 abr 2013.

FUTURE INTERNET RESEARCH AND EXPERIMENTATION- FIRE. **European commission:** future internet research and experimentation, 2008. Disponível em: <https://www.ict-fire.eu/projects>. Acesso em: 18 abr 2013.

FLOODLIGHT. **The floodlight controller.** Big Switch Networks: Project Floodlight, 2013. Disponível em: < http://www.projectfloodlight.org>. Acesso em: 25 abr 2013.

GENI. **Global environment for network innovations**. Estados Unidos: National Science Foundation, 2016. Disponível em: <http://www.geni.net>. Acesso em: 29 maio 2013.

GENI. **Global environment for network innovations:** spiral 2 overview. Estados Unidos: National Science Foundation, 2010. Disponível em: <http://groups.geni.net/geni/attachment/wiki/SpiralTwo/GENISysOvrvw092908.pdf>. Acesso em: 10 abr 2013.

GIGA. **Rede experimental de alta velocidade**. Campinas: CPqD-RNP, 2011. Disponível em: <http://www.giga.org.br/rede/infra>. Acesso em: 10 abr 2013.

GREENBERG A.; HAMILTON J.; MALTZ D.; PATEL P. The cost of a cloud: research problems in data center networks. In: SIGCOMM COMPUTER COMMUNICATION REVIEW- SIGCOMM, 9., 2009, Barcelona. **Proceedings...** Barcelona: ACM, 2009. p. 68–73.

GRUNWALD, D.; LEVIS, P.; FARKAS, K.; NEUFELD, M. Policies for dynamic clock scheduling. In: CONFERENCE ON SYMPOSIUM ON OPERATING SYSTEM DESIGN & IMPLEMENTATION- OSDI, 4., 2000, San Diego. **Proceedings...** San Diego: USENIX Association, 2000. p. 6–6.

GUEDES, D.. Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. In: MINICURSOS DO SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES- SBRC, 30, 2012, Ouro Preto. **Anais...** Ouro Preto: Universidade Federal de Minas Gerais, 2012. v. 30, p. 160–210.

HELLER, B.; SEETHARAMAN, S.; MAHADEVAN, P.; YIAKOUMIS, Y.; SHARMA, P.; BANERJEE, S.; MCKEOWN, N. ElasticTree: saving energy in data center networks. In: USENIX CONFERENCE ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION-NSDI, 7., 2010, San Jose. **Proceedings...** San Jose: USENIX Association, 2010. p.17–33.

HUONG, T. T.; NAM, P. N.; THANH, N. H.; SCHLOSSER, D.; JARSCHEL, M.; PRIES, R. ECODANE – Reducing energy consumption in data center networks based on traffic engineering. In: WÜRZBURG WORKSHOP ON IP: JOINT ITG AND EURO-NF WORKSHOP VISIONS OF FUTURE GENERATION NETWORKS, 11., 2011, Würzburg. **Proceedings...** Würzburg: EuroView, 2011. p. 87–88.

INTERNACIONAL ENERGY AGENCY- IEA. **World:** balances for 2013. France: OECD/IEA, 2013. Disponível em: <http://www.iea.org/statistics/statisticssearch/report/?year=2013&country=WORLD&product=Balances>. Acesso em: 20 nov. 2014.

JAIN, R. Internet 3.0: ten problems with current internet architecture and solutions for the next generation. In: PROCEEDINGS OF THE; IEEE CONFERENCE ON MILITARY COMMUNICATIONS - MILCOM, 6., 2006, Piscataway. **Proceedings...** Piscataway: IEEE, 2006. p. 153–161.

JARSCHEL, M.; PRIES, R. An openflow-based energy-efficient data center approach. In: CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION- ACM SIGCOMM, 12., 2012, Helsinki. **Proceedings…** Helsinki: ACM, 2012. p. 13–17.

JOSEPH, W.; LEWIS, C. **Green:** the new computing coat of arms? **IT Professional**, Japan, v. 10, n. 1, p. 1216, 2008.

MATTOS, D. M. F. **XenFlow:** um sistema de processamento de fluxos robusto e eficiente para redes virtuais. Rio de Janeiro: Escola Politécnica,2011. 71 p. Projeto Engenharia de Computação e Informação.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: enabling innovation in campus networks. In: PROCEEDINGS OF THE ACM SIGCOMM COMPUTER COMMUNICATION REVIEW, 8., 2008, Seattle. **Proceedings…** Seattle: ACM, 2008. p. 69–74.

MCKEOWN, N. Openflow: enabling innovation in campus networks. **Computer Communication Review,** New York, v.38, n. 2, p. 69–74, 2008.

MININET. **An instant virtual network on your laptop (or other PC)**. Estados Unidos: Universidade de Stanford, 2016. Disponível em: <http://mininet.org>. Acesso em: 21 jan. 2016.

MOGUL, C. Devoflow: cost-eective flow management for high performance enterprise networks. In: SIGCOMM WORKSHOP ON HOT TOPICS IN NETWORKS, 9., 2010, Hotnets. **Proceedings...** Hotnets: ACM, 2010. v. 10, p. 1–6.

MOREIRA, M.; FERNANDES, N.; COSTA L.; DUARTE O. Internet do futuro: um novo horizonte. In: MINICURSOS DO SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES-SBRC, 27., 2009, Recife. **Anais...** Recife: Sociedade Brasileira de Computação, 2009. p. 1–59.

MOSKOWITZ, R.; NIKANDER, P.; JOKELA, P.; HENDERSON, T. **Host identity protocol**. Fremont: IETF,2005. p. 1. (RFC, 5201).

NASCIMENTO, M. R.; ROTHENBERG, C. E.; SALVADOR, M. R.; MAGALHÃES, M. F. Quagflow: partnering quagga with openflow. In: SIGCOMM CONFERENCE ON SIGCOMM, SIGCOMM, 10., 2010, New York. **Proceedings...** New York**:** ACM, 2010. p. 441–442.

OFELIA. **Openflow in europe:** linking infrastructure and applications. Berlin: European Center for Information and Communication Technologies, 2011. Disponível em: <http://www.fp7-ofelia.eu>. Acesso em: 20 abr 2013.

CONTROL AND MANAGEMENT FRAMEWORK- OMF. **OMF:** a control and management framework for networking testbeds. Australia: Rutgers University,2011. Disponível em: <http://omf.mytestbed.net>. Acesso em: 20 fev. 2013.

OPENFLOW. **Basic spanning tree**. Estados Unidos: Universidade de Stanford, 2011. Disponível em: <http://archive.openflow.org/wk/index.php/Basic_Spanning_Tree>. Acesso em: 23 maio 2015.

PATEL, C. D.; BASH, C. E.; SHARMA, R.; BEITELMAM, M.; FRIEDRICH, R. Smart cooling of data centers. In: INTERNATIONAL ELECTRONIC PACKAGING TECHNICAL CONFERENCE AND EXHIBITION, 2., 2003, Maui. **Proceedings...** Maui: ASME, 2003. p. 129–137.

PAUL, S.; PAN, J.; JAIN, R. Architectures for the future networks and the next generation internet: a survey. **Comput. Commun.,** Netherlands, v. 34, n. 1, p. 2–42, 2011.

PETERSON, L.; MUIR, S.; ROSCOE, T.; KLINGAMAN, A. **Planetlab architecture:** an overview. Berkeley: Princeton University, 2006. Disponível em: <http://www.planet-lab.org/files/pdn/PDN-06-031/pdn-06-031.pdf>. Acesso em: 20 maio 2013.

PETERSON, L.; SEVINC, S.; LEPREAU, J.; RICCI, R.; WROCLAWSKI, J.; FABER, T.; SCHWAB, S.; BAKER, S. **Slice based facility architecture, draft version 1.04**. Campo Grande: Sociedade Brasileira de Computação, 2009. Disponível em: <http://svn.planetlab.org/attachment/wiki/GeniWrapper/sfa.pdf>. Acesso em: 3 abr 2013.

PROJECT FLOODLIGHT. **Open source software for building software-defined networks**. Big Switch Networks: Project Floodlight, 2015. Disponível em: <http://www.projectfloodlight.org>. Acesso em: 11 jun. 2015.

PROTOGENI. Estados Unidos: University of Utah, 2011. Disponível em: <http://groups.geni.net/geni/wiki/ProtoGENI>. Acesso em: 20 mar 2013.

RAM, K. K. sNICh: efficient last hop networking in the data center. In: SYMPOSIUM ON ARCHITECTURES FOR NETWORKING AND COMMUNICATIONS SYSTEMS-ANCS, 6., 2010, San Diego. **Proceedings...** San Diego: IEEE, 2010. v. 10, p. 1–26.

RIEKSTIN, A. C.; JANUÁRIO, G. C.; RODRIGUES, B. B.; NASCIMENTO, V. T.; CARVALHO, T. C. M. B.; MEIROSU, C. Orchestration of energy efficiency capabilities in networks. **Journal of Network and Computer Applications,** Oklahoma, v. 59, p. 74–87, Jan. 2016.

ROTHENBERG, C. E.; NASCIMENTO, M. R.; SALVADOR, M. R.; MAGALHÃES M. F. Openflow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. **Cad. CPqD Tecnologia,** Campinas, v. 7, n. 1, p. 65–76, 2011.

SIVARAMAN, V.; REVIRIEGO ZHAO, P. Z.; SÁNCHEZ-MACIÁN, A.; VISHWANATH, A.; MAESTRO, J. A.; RUSSELL, C. An experimental power profile of energy efficient ethernet switches. **Comput. Commun**., Piza, v. 50, p. 110–118, Sep. 2014.

SCARABUCCI, R. R.; STANTON, M. A.; BARROS, M. R. X.; SALVADOR, M. R.; ROSSI, S. M.; SIM, F. D.; ROCHA, M. L.; SILVA NETO, I. L.; ROSOLEM, J. B.; FUDOLI, T. R. T.; MENDES, J. M. D.; CASTRO, N. F.; MACHADO, I.; REGGIANI, A. E.; PARADISI, A.; MARTINS, L. Project giga-high-speed experimental ip/wdm network. In: INTERNATIONAL CONFERENCE ON TESTBEDS AND RESEARCH INFRASTRUCTURES FOR THE DEVELOPMENT OF NETWORKS & COMMUNITIES, 1., 2005, Trento. **Proceedings…** Trento: IEEE, 2005. p. 242–251.

SHARMA, S.; STAESSENS, D.; COLLE, D.; PICKAVET, M.; DEMEESTER, P. Openflow: meeting carrier-grade recovery requirements. **Comput. Commun.,** Netherlands, v. 36, n. 6, p. 656–665, Mar. 2013.

SILVA, M. R. P.; ZANETI, G. B.; ZAGO, M. G. TI verde: princípios e práticas sustentáveis para aplicação em universidades. In: SIMPÓSIO BRASILEIRO DE SISTEMAS ELÉTRICOS,3., 2010, Belém, 2010. **Anais…** Belém: Universidade Federal do Pará, 2010. p. 1–6.

STOICA, I.; ADKINS, D.; ZHUANG, S.; SHENKER, S.; SURANA, S. Internet indirection infrastructure. **IEEE/ACM Trans. Netw.,** New York, v. 12, n. 4, p. 205–218, 2004.

THANH, N. H.; NAM, P. N.; HUONG, T. T.; HUNG N. T.; DOANH, L. K.; PRIES R.. Enabling experiments for energy-efficient data center networks on openFlow-based platform. In: INTERNATIONAL CONFERENCE ON COMMUNICATIONS AND ELECTRONICS, 4., 2012, Hue. **Proceedings...** Hue: ICCE, 2012. p. 239–244.

WANG, Y.; KELLER E.; BISKEBORN, B.; MERWE, J. V. D; REXFORD, J. Virtual routers on the move: live router migration as a network-management primitive. **Computer Communication Review**, Seattle, v. 38, n. 4, p. 231–242, 2008.

WANG, X.; YAO, Y.; WANG, X.; LU, K.; CAO, Q. CARPO: Correlation-aware power optimization in data center networks. In: IEEE INFOCOM, 2012, Orlando. **Proceedings...** Orlando: IEEE, 2012. p. 1125–1133.

WANG, H.; LI, Y.; JIN, D.; HUI, P.; WU, J. Saving energy in partially deployed software defined networks. **IEEE Trans. Comput**., Torino, v. 65, n. 5, p. 1578–1592, May 2016.

WEBSCIENCE. **Brazilian institute for web science research**. Rio de Janeiro: INCT, 2010. Disponível em: <http://webscience.org.br/files/INCTportugues2010.pdf>. Acesso em: 10 maio 2013.

XIE, J.; GUO, D.; HU, Z.; QU, T.; LV, P. Control plane of software defined networks: a survey. **Comput. Commun.**, Pisa, v. 67, [s.n], p. 1–10, Aug. 2015.

APÊNDICES

APÊNDICE A - Algoritmo do Módulo Economia de Energia no controlador Floodlight

Algoritmo do Módulo Economia de Energia

```
1  InicioAlgoritmo
2  Pacotes LLDP são enviados do Módulo Link Discovery para o Módulo Topology
   Manager;
3  Módulo Topology Manager envia pacotes LLDP para os switches;
4  Switches retornam informações de conectividade (pacotes LLDP) para o Módulo
   Topology Manager;
5  Módulo Topology Manager encaminha as informações de conectividade para o Módulo
   Link Discovery;
6  Enquanto (Módulo Link Discovery registra mudança) Faca
7      Lista das conexões são atualizadas (ligações ativadas, desativadas, com valores de
       custos, etc.);
8      Mensagem de notificação (linkUpdate) é enviada ao Módulo Economia de Energia;
9      Modulo Economia Energia recebe:{MAC switch origem,
                                       Porta Ethernet do switch de origem,
                                       Estado da porta do switch de origem,
                                       MAC switch de destino,
                                       Porta Ethernet do switch de destino,
                                       Estado da porta do switch de destino,
                                       Valor de custo entre as ligações dos switches};
10 FimEnquanto
11 Cálculo da Árvore de Extensão Mínima sobre os switches OpenFlow que não formem
   ciclos entre os seus vértices (Algoritmo MiNet);
12 Conjunto de mudanças mantém a lista de ligações que mudaram com a árvore;
13 Enquanto (switches estão dentro da Árvore de Extensão Mínima) Faça
14     Portas Ethernet dos switches recebem mensagens OpenFlow para modo ativado;
15 FimEnquanto
16 Enquanto (switches estão fora da Árvore de Extensão Mínima) Faça
17     Portas Ethernet dos switches recebem mensagens OpenFlow para modo desativado;
18     Se (fluxo de dados é excessivo) Entao
19             Algumas ou todas as portas Ethernet dos switches recebem mensagens
               OpenFlow para modo ativado;
20     FimSe
21 FimEnquanto
22 FimAlgoritmo
```

APÊNDICE B - Arquivo de propriedades “LinkCustoInicial.properties”

O arquivo de propriedades “LinkCustoInicial.properties” com os custos iniciais nas ligações dos *switches* se encontra no caminho /floodlight-1.0/src/main/resources/

FatTree10

```
1,3=10
1,4=10
1,7=10
1,8=10
2,3=10
2,4=10
2,7=10
2,8=10
3,5=8
3,6=8
4,5=8
4,6=8
7,9=8
7,10=8
8,9=8
8,10=8
```

FatTree20

```
1,5=10
 1,9=10
 1,13=10
 1,17=10
 2,5=10
 2,9=10
 2,13=10
 2,17=10
 3,6=10
 3,10=10
 3,14=10
 3,18=10
 4,6=10
 4,10=10
 4,14=10
 4,18=10
 5,7=8
 5,8=8
 6,7=8
 6,8=8
 9,11=8
 9,12=8
 10,11=8
 10,12=8
 13,15=8
 13,16=8
 14,15=8
 14,16=8
 17,19=8
 17,20=8
 18,19=8
 18,20=8
```

APÊNDICE C - Arquivo de script “LinkNovoCusto.sh”

O arquivo de script “LinkNovoCusto.sh” com novos custos nas ligações dos *switches* se encontra no caminho /floodlight-1.0/src/main/resources/

FatTree10

```
#!/bin/bash

CUSTO="[{
\"1,3\":16,\"1,4\":16,\"1,7\":16,\"1,8\":16,\"2,3\":14,\"2,4\":14,\"2,7\":14,\"2,8\":14,\"3,5\":1
2,\"3,6\":12,\"4,5\":12,\"4,6\":12,\"7,9\":10,\"7,10\":10,\"8,9\":10,\"8,10\":10}]"

curl -d $ CUSTO http://$127.0.0.1:8080/wm/ecoenergia/novocusto/json
```

FatTree20

```
#!/bin/bash

CUSTO="[{\"1,5\":18,\"1,9\":18,\"1,13\":18,\"1,17\":18,\"2,5\":14,\"2,9\":14,\"2,13\":14,\"2
,17\":14,\"3,6\":16,\"3,10\":16,\"3,14\":16,\"3,18\":16,\"4,6\":12,\"4,10\":12,\"4,14\":12,\"4,
18\":12,\"5,7\":16,\"5,8\":16,\"6,7\":16,\"6,8\":16,\"9,11\":12,\"9,12\":12,\"10,11\":12,\"10,1
2\":12,\"13,15\":14,\"13,16\":14,\"14,15\":14,\"14,16\":14,\"17,19\":10,\"17,20\":10,\"18,19\
":10,\"18,20\":10}]"

curl -d $ CUSTO http://$127.0.0.1:8080/wm/ecoenergia/novocusto/json
```

APÊNDICE D - Algoritmo MiNet

Algoritmo MiNet	
1	**InicioAlgoritmo**
2	Vetor da topologia dos switches (grafo da rede) com custos nos links;
3	Escolha randomicamente de um switch pelo algoritmo ou escolha manual pelo administrador;
4	**Enquanto** (“Switches fora do subgrafo” <> vazio) **Faça**
5	Identifique o switch mais próximo (menor custo) e sem ciclo do (s) switch (es) selecionado (s) anteriormente no conjunto “Possíveis ligações com menor custo e sem ciclos”;
6	Inclua este switch e a respectiva ligação no conjunto “Ligações incluídas no subgrafo”;
7	Os switches da Árvore de Extensão Mínima formam o conjunto “Switches dentro do subgrafo” e os switches restantes formam o conjunto “Switches fora do subgrafo”;
8	Inclua no conjunto “Possíveis ligações com menor custo e sem ciclos” o (s) switch (es) do conjunto “Switches dentro do subgrafo” que tenham conexão com o (s) switch (es) do conjunto “Switches fora do subgrafo”;
9	Repita a linha 5 até que todos os switches estejam conectados, ou seja, o conjunto “Switches fora do subgrafo” esteja vazio;
10	**FimEnquanto**
11	**FimAlgoritmo**

APÊNDICE E - API REST

A API REST é uma interface recomendada para desenvolver aplicações que utilizam recursos suportados pelo Floodlight. O REST mostra informações de um *switch* em uso, para recuperar os dados dos *switches* conectados ao Floodlight, tem-se duas formas: 1) interface do usuário no Floodlight ou a chamada 2) API REST.

1) Com a interface do usuário no Floodlight, acessando pelo navegador de Internet na URL:

```
http://127.0.0.1:8080/ui/index.html
```

2) Com a API REST, no console do Linux, acessando com o comando “curl”, por exemplo:

```
curl http://127.0.0.1:8080/wm/core/controller/switches/json
```

A seguir serão listadas as RESTs para verificar as informações coletadas do Módulo Economia de Energia.

a) REST “switches” para exibir os endereços MAC dos switches conectados ao controlador:

```
curl http://127.0.0.1:8080/wm/core/controller/switches/json
```

b) REST “custolink” para exibir os custos nas ligações entre os switches:

```
curl http://127.0.0.1:8080/wm/ecoenergia/custolink/json
```

c) REST “switchlink” para exibir o custo, *switch* de origem, porta Ethernet de origem, *switch* de destino e porta Ethernet de destino de cada *switch* conectado no Floodlight:

```
curl http://127.0.0.1:8080/wm/ecoenergia/switchlink/json
```

d) REST “mstlink” para exibir o custo, *switch* de origem, porta Ethernet de origem, *switch* de destino e porta Ethernet de destino, somente dos *switches* e das respectivas portas Ethernet que se encontram ligadas:

```
curl http://127.0.0.1:8080/wm/ecoenergia/mstlink/json
```

e) REST “ociosolink” para exibir o custo, *switch* de origem, porta Ethernet de origem, *switch* de destino e porta Ethernet de destino, somente dos *switches* e das respectivas portas Ethernet que se encontram ociosas e desligadas:

```
curl http://127.0.0.1:8080/wm/ecoenergia/ociosolink/json
```

APÊNDICE F - Arquivo de configuração “floodlight.properties”

O arquivo de configuração “floodlight.properties” modificado no controlador Floodlight se encontra no caminho /floodlight-1.0/src/main/resources/

```
floodlight.modules=\
net.floodlightcontroller.jython.JythonDebugInterface,\
net.floodlightcontroller.counter.CounterStore,\
net.floodlightcontroller.storage.memory.MemoryStorageSource,\
net.floodlightcontroller.core.internal.FloodlightProvider,\
net.floodlightcontroller.threadpool.ThreadPool,\
net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl,\
net.floodlightcontroller.devicemanager.internal.DefaultEntityClassifier,\
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher,\
net.floodlightcontroller.firewall.Firewall,\
net.floodlightcontroller.forwarding.Forwarding,\
net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager,\
net.floodlightcontroller.topology.TopologyManager,\
net.floodlightcontroller.flowcache.FlowReconcileManager,\
net.floodlightcontroller.debugcounter.DebugCounter,\
net.floodlightcontroller.debugevent.DebugEvent,\
net.floodlightcontroller.perfmon.PktInProcessingTime,\
net.floodlightcontroller.ui.web.StaticWebRoutable,\
net.floodlightcontroller.loadbalancer.LoadBalancer,\
org.sdnplatform.sync.internal.SyncManager,\
net.floodlightcontroller.devicemanager.internal.DefaultEntityClassifier,\
net.floodlightcontroller.ecoenergia.EcoEnergia
org.sdnplatform.sync.internal.SyncManager.authScheme=CHALLENGE_RESPONSE
org.sdnplatform.sync.internal.SyncManager.keyStorePath=/etc/floodlight/auth_credentials.j
ceks
org.sdnplatform.sync.internal.SyncManager.dbPath=/var/lib/floodlight/
net.floodlightcontroller.restserver.RestApiServer.port = 8080
net.floodlightcontroller.core.FloodlightProvider.openflowport = 6633
net.floodlightcontroller.jython.JythonDebugInterface.port = 6655
net.floodlightcontroller.forwarding.Forwarding.idletimeout = 5
net.floodlightcontroller.forwarding.Forwarding.hardtimeout = 0
```

APÊNDICE G - Arquivo "net.floodlightcontroller.core.module.IFloodlightModule"

O arquivo do módulo "net.floodlightcontroller.core.module.IFloodlightModule" modificado no controlador Floodlight se encontra no caminho /floodlight-1.0/src/main/resources/META-INFO/services/

```
net.floodlightcontroller.core.module.ApplicationLoader
net.floodlightcontroller.core.internal.FloodlightProvider
net.floodlightcontroller.storage.memory.MemoryStorageSource
net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl
net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager
net.floodlightcontroller.topology.TopologyManager
net.floodlightcontroller.forwarding.Forwarding
net.floodlightcontroller.flowcache.FlowReconcileManager
net.floodlightcontroller.core.OFMessageFilterManager
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher
net.floodlightcontroller.perfmon.PktInProcessingTime
net.floodlightcontroller.perfmon.NullPktInProcessingTime
net.floodlightcontroller.restserver.RestApiServer
net.floodlightcontroller.learningswitch.LearningSwitch
net.floodlightcontroller.hub.Hub
net.floodlightcontroller.jython.JythonDebugInterface
net.floodlightcontroller.counter.CounterStore
net.floodlightcontroller.counter.NullCounterStore
net.floodlightcontroller.debugcounter.DebugCounter
net.floodlightcontroller.debugevent.DebugEvent
net.floodlightcontroller.threadpool.ThreadPool
net.floodlightcontroller.ui.web.StaticWebRoutable
net.floodlightcontroller.virtualnetwork.VirtualNetworkFilter
net.floodlightcontroller.firewall.Firewall
net.floodlightcontroller.loadbalancer.LoadBalancer
org.sdnplatform.sync.internal.SyncManager
org.sdnplatform.sync.internal.SyncTorture
net.floodlightcontroller.devicemanager.internal.DefaultEntityClassifier
net.floodlightcontroller.ecoenergia.EcoEnergia
```

APÊNDICE H - Arquivo de depuração de mensagens “logback.xml”

O arquivo de depuração de mensagens “logback.xml” modificado no controlador Floodlight se encontra no caminho /floodlight-1.0/src/main/resources/META-INFO/services/

```
<configuration scan="true">
<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
<encoder>
<pattern>%level [%logger:%thread] %msg%n</pattern>
</encoder>
</appender>
<root level="INFO">
<appender-ref ref="STDOUT" />
</root>
<logger name="org" level="WARN"/>
<logger name="LogService" level="WARN"/><!-- Restlet access logging -->
<logger name="net.floodlightcontroller" level="INFO"/>
<logger name="net.floodlightcontroller.logging" level="WARN"/>
<logger name=" net.floodlightcontroller" level="DEBUG"/>
<logger name=" net.floodlightcontroller.ecoenergia.EcoEnergia" level="DEBUG"/>
<logger name=" net.floodlightcontroller.ecoenergia.algoritmo" level="DEBUG"/>
<logger name=" net.floodlightcontroller.ecoenergia.tipos.ComparacaoLink" level="INFO"/>
<logger name="org.sdnplatform" level="INFO"/>
</configuration>
```

APÊNDICE I - Geração de tráfego com D-ITG

Na janela do terminal do *host* h1, é executada a **Instrução 1** para informar que h1 será o servidor e receberá o tráfego de dados.

Instrução 1 no *host* h1

```
$ ./ITGRecv
```

Na janela do terminal do *host* h2, é executada a **Instrução 2** para informar que h2 será o cliente e enviará o tráfego de dados. O mesmo será feito para os demais *hosts* da topologia FatTree10 (h3, h4, h5, h6, h7 e h8) e da topologia FatTree20 (h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, h13, h14, h15 e h16) enviando o tráfego para h1.

Instrução 2 no *host* h2

```
$ ./ITGSend script_fluxos –l emissorh2.log –x receptorh1.log
```

O arquivo **script_fluxos** contém a informação abaixo para gerar dez fluxos do protocolo TCP (-T TCP), para o endereço IP de destino do h1 (-a 10.10.1.34), com tamanho do pacote constante de 512 bytes (-c 512), taxa constante de 100 pps (-C 100) e com duração de tempo por 30.000 ms (-t 30000)

script_fluxos:

```
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
-T TCP -a 10.10.1.34 -c 512 -C 100 -t 30000
```

Dois arquivos de registros a nível de pacote serão criados tanto na opção do remetente (–l emissorh2.log) como do lado da recepção (–x receptorh1.log)

Exemplos de resultados com os arquivos de registros gerados

No **Resultado 1** é apresentado o registro do Total de Resultados obtidos dos dez fluxos contidos no arquivo emissorh2.log. Esse registro é emitido com a **Instrução 3**.

Instrução 3

```
$ ./ITGDec emissorh2.log
```

Resultado 1 - emissorh2.log

```
_____________________________________________________________

**********  TOTAL RESULTS  **********

_____________________________________________________________

Number of flows         =         10
Total time              =   30.085752  s
Total packets           =       9938
Minimum delay           =   0.000000  s
Maximum delay           =   0.000000  s
Average delay           =   0.000000  s
Average jitter          =   0.000000  s
Delay standard deviation =  0.000000  s
Bytes received          =    5088256
Average bitrate         =  1353.000849  Kbit/s
Average packet rate     =   330.322473  pkt/s
Packets dropped         =          0  (0.00 %)
Average loss-burst size =          0  pkt
Error lines             =          0

-------------------------------------------------------------------------------
```

Do mesmo modo, no **Resultado 2** é apresentado o registro do Total de Resultados obtidos dos dez fluxos contidos no arquivo receptorh1.log. Esse registro é emitido com a **Instrução 4**.

Instrução 4

```
$ ./ITGDec receptorh1.log
```

Resultado 2 - receptorh1.log

```
_____________________________________________________________________

**********  TOTAL RESULTS  **********

_____________________________________________________________________

Number of flows         =         10
Total time              =   40.073818  s
Total packets           =       9938
Minimum delay           =   -0.000488  s
Maximum delay           =   11.608000  s
Average delay           =    3.529225  s
Average jitter          =    0.093486  s
Delay standard deviation =    1.926007  s
Bytes received          =    5088256
Average bitrate         = 1015.776635  Kbit/s
Average packet rate     =  247.992343  pkt/s
Packets dropped         =          0  (0.00 %)
Average loss-burst size =          0  pkt
Error lines             =          0
---------------------------------------------------------------------------
```

APÊNDICE J - Topologia com 10 *switches*: configurações nos componentes de rede no rack OpenFlow InstaGENI da Universidade de UtahDDC.

Node #1

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s9	pc7	2016-12-31T23:59:59.000Z	emulab-xen	s9.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 33087 ssh ligiapre@pc7.utahddc.geniracks.net -p 33087				
Interfaces		**MAC**		**Layer 3**	
interface-26	pc7:lo0	022ecf10ef4c		ipv4: 10.10.1.26	
interface-30	pc7:lo0	023bec2fec44		ipv4: 10.10.1.30	
interface-42	pc7:eth1	02eff595a2a0		ipv4: 10.10.1.42	
interface-44	pc7:eth1	02b271fae24b		ipv4: 10.10.1.44	
interface-66	pc7:lo0	02e45292ec90		ipv4: 10.10.1.66	

Node #2

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s1	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s1.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 33082 ssh ligiapre@pc10.utahddc.geniracks.net -p 33082				
Interfaces		**MAC**		**Layer 3**	
interface-1	pc10:eth1	02771b49a5c7		ipv4: 10.10.1.1	
interface-3	pc10:eth1	02202638f7ba		ipv4: 10.10.1.3	
interface-6	pc10:eth1	020ab4c24633		ipv4: 10.10.1.6	
interface-8	pc10:eth1	02fef92e4cc2		ipv4: 10.10.1.8	
interface-50	pc10:eth1	0204adcbce97		ipv4: 10.10.1.50	

Node #3

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s2	pc7	2016-12-31T23:59:59.000Z	emulab-xen	s2.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 33084 ssh ligiapre@pc7.utahddc.geniracks.net -p 33084				
Interfaces		**MAC**		**Layer 3**	
interface-9	pc7:eth1	02c94e32e99d		ipv4: 10.10.1.9	
interface-11	pc7:eth1	028b71162e84		ipv4: 10.10.1.11	
interface-13	pc7:lo0	02b42fb2a7a1		ipv4: 10.10.1.13	
interface-15	pc7:lo0	021bb103ae54		ipv4: 10.10.1.15	
interface-52	pc7:lo0	02ec770b02fa		ipv4: 10.10.1.52	

Node #4

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s3	pc3	2016-12-31T23:59:59.000Z	emulab-xen	s3.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc3.utahddc.geniracks.net -p 33082 ssh ligiapre@pc3.utahddc.geniracks.net -p 33082				
Interfaces		**MAC**		**Layer 3**	
interface-2	pc3:eth1	02385d2fcf95		ipv4: 10.10.1.2	
interface-10	pc3:eth1	02262ca9c53e		ipv4: 10.10.1.10	
interface-17	pc3:eth1	028cbb2b266e		ipv4: 10.10.1.17	
interface-19	pc3:eth1	02c44a60a304		ipv4: 10.10.1.19	
interface-54	pc3:eth1	026766713dba		ipv4: 10.10.1.54	

Node #5

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s4	pc11	2016-12-31T23:59:59.000Z	emulab-xen	s4.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc11.utahddc.geniracks.net -p 33082 ssh ligiapre@pc11.utahddc.geniracks.net -p 33082				

Interfaces		MAC	Layer 3
interface-4	pc11:eth1	0299cb03e784	ipv4: 10.10.1.4
interface-12	pc11:eth1	02a8a7195b84	ipv4: 10.10.1.12
interface-21	pc11:eth1	0234abd78d0e	ipv4: 10.10.1.21
interface-23	pc11:eth1	024116b29cde	ipv4: 10.10.1.23
interface-56	pc11:eth1	020533ac1ce8	ipv4: 10.10.1.56

Node #6

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s5	pc6	2016-12-31T23:59:59.000Z	emulab-xen	s5.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc6.utahddc.geniracks.net -p 33082 ssh ligiapre@pc6.utahddc.geniracks.net -p 33082				

Interfaces		MAC	Layer 3
interface-18	pc6:eth1	0201e8cdd120	ipv4: 10.10.1.18
interface-22	pc6:eth1	021b7763bf75	ipv4: 10.10.1.22
interface-33	pc6:eth1	0252c95837e3	ipv4: 10.10.1.33
interface-35	pc6:eth1	02979aea059d	ipv4: 10.10.1.35
interface-62	pc6:eth1	022871c2ba13	ipv4: 10.10.1.62

Node #7

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s6	pc2	2016-12-31T23:59:59.000Z	emulab-xen	s6.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc2.utahddc.geniracks.net -p 33082 ssh ligiapre@pc2.utahddc.geniracks.net -p 33082				

Interfaces		MAC	Layer 3
interface-20	pc2:eth1	029135dcdd3b	ipv4: 10.10.1.20
interface-24	pc2:eth1	029b5b861b9e	ipv4: 10.10.1.24
interface-37	pc2:eth1	021fadfbfd58	ipv4: 10.10.1.37
interface-39	pc2:eth1	0207caaf355f	ipv4: 10.10.1.39
interface-64	pc2:eth1	022b865f6df6	ipv4: 10.10.1.64

Node #8

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s7	pc7	2016-12-31T23:59:59.000Z	emulab-xen	s7.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 33085 ssh ligiapre@pc7.utahddc.geniracks.net -p 33085				

Interfaces		MAC	Layer 3
interface-5	pc7:eth1	02f67f8c5251	ipv4: 10.10.1.5
interface-14	pc7:lo0	028ac2b45941	ipv4: 10.10.1.14
interface-25	pc7:lo0	021dec006be1	ipv4: 10.10.1.25
interface-27	pc7:lo0	02a197f7b2cf	ipv4: 10.10.1.27
interface-58	pc7:lo0	024d46c87db0	ipv4: 10.10.1.58

Node #9

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s8	pc7	2016-12-31T23:59:59.000Z	emulab-xen	s8.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 33086 ssh ligiapre@pc7.utahddc.geniracks.net -p 33086				
Interfaces		**MAC**		**Layer 3**	
interface-7	pc7:eth1	0228e18c7fba		ipv4: 10.10.1.7	
interface-16	pc7:lo0	0270e2fa776c		ipv4: 10.10.1.16	
interface-29	pc7:lo0	02cc2e93d7b7		ipv4: 10.10.1.29	
interface-31	pc7:lo0	021b4ec56fd6		ipv4: 10.10.1.31	
interface-60	pc7:lo0	0234fbe8cdcf		ipv4: 10.10.1.60	

Node #10

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s10	pc7	2016-12-31T23:59:59.000Z	emulab-xen	s10.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 33083 ssh ligiapre@pc7.utahddc.geniracks.net -p 33083				
Interfaces		**MAC**		**Layer 3**	
interface-28	pc7:lo0	0270091dc76f		ipv4: 10.10.1.28	
interface-32	pc7:lo0	020af9c66c57		ipv4: 10.10.1.32	
interface-46	pc7:eth1	02edeb038cc7		ipv4: 10.10.1.46	
interface-48	pc7:eth1	02f693cc278f		ipv4: 10.10.1.48	
interface-67	pc7:lo0	02e36b3ca9d5		ipv4: 10.10.1.67	

Node #11

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h1	pc13	2016-12-31T23:59:59.000Z	emulab-xen	h1.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc13.utahddc.geniracks.net -p 33082 ssh ligiapre@pc13.utahddc.geniracks.net -p 33082				
Interfaces		**MAC**		**Layer 3**	
interface-34	pc13:eth1	02a0f4bc17ae		ipv4: 10.10.1.34	

Node #12

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h2	pc14	2016-12-31T23:59:59.000Z	emulab-xen	h2.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc14.utahddc.geniracks.net -p 33082 ssh ligiapre@pc14.utahddc.geniracks.net -p 33082				
Interfaces		**MAC**		**Layer 3**	
interface-36	pc14:eth1	0255ba34bb98		ipv4: 10.10.1.36	

Node #13

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h3	pc4	2016-12-31T23:59:59.000Z	emulab-xen	h3.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc4.utahddc.geniracks.net -p 33082 ssh ligiapre@pc4.utahddc.geniracks.net -p 33082				
Interfaces		**MAC**		**Layer 3**	
interface-38	pc4:eth1	02326fe355ff		ipv4: 10.10.1.38	

Node #14

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h4	pc1	2016-12-31T23:59:59.000Z	emulab-xen	h4.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc1.utahddc.geniracks.net -p 33082 ssh ligiapre@pc1.utahddc.geniracks.net -p 33082				
Interfaces		MAC		Layer 3	
interface-40	pc1:eth1	023b0da159c9		ipv4: 10.10.1.40	

Node #15

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h5	pc8	2016-12-31T23:59:59.000Z	emulab-xen	h5.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc8.utahddc.geniracks.net -p 33082 ssh ligiapre@pc8.utahddc.geniracks.net -p 33082				
Interfaces		MAC		Layer 3	
interface-41	pc8:eth1	022e50da4d04		ipv4: 10.10.1.41	

Node #16

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h6	pc5	2016-12-31T23:59:59.000Z	emulab-xen	h6.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc5.utahddc.geniracks.net -p 33082 ssh ligiapre@pc5.utahddc.geniracks.net -p 33082				
Interfaces		MAC		Layer 3	
interface-43	pc5:eth1	02fb0d055ee9		ipv4: 10.10.1.43	

Node #17

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h7	pc12	2016-12-31T23:59:59.000Z	emulab-xen	h7.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc12.utahddc.geniracks.net -p 33082 ssh ligiapre@pc12.utahddc.geniracks.net -p 33082				
Interfaces		MAC		Layer 3	
interface-45	pc12:eth1	023f63ac35f7		ipv4: 10.10.1.45	

Node #18

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h8	pc15	2016-12-31T23:59:59.000Z	emulab-xen	h8.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 33082 ssh ligiapre@pc15.utahddc.geniracks.net -p 33082				
Interfaces		MAC		Layer 3	
interface-47	pc15:eth1	02ef33abce98		ipv4: 10.10.1.47	

Node #19

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	controller	pc7	2016-12-31T23:59:59.000Z	emulab-xen	controller.FatTree8.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 33082 ssh ligiapre@pc7.utahddc.geniracks.net -p 33082				
Interfaces		MAC		Layer 3	
interface-53	pc7:eth1	02f47704113c		ipv4: 10.10.1.53	
interface-49	pc7:eth1	02ab2dac858d		ipv4: 10.10.1.49	
interface-61	pc7:eth1	02e2ac945a4b		ipv4: 10.10.1.61	
interface-55	pc7:eth1	0237e516a367		ipv4: 10.10.1.55	
interface-63	pc7:eth1	02eeb47f03f6		ipv4: 10.10.1.63	
interface-51	pc7:lo0	023d7574f39c		ipv4: 10.10.1.51	
interface-57	pc7:lo0	02874a4f40f3		ipv4: 10.10.1.57	
interface-59	pc7:lo0	023386297a5b		ipv4: 10.10.1.59	
interface-65	pc7:lo0	02762f10b231		ipv4: 10.10.1.65	
interface-68	pc7:lo0	02f88f800f40		ipv4: 10.10.1.68	

APÊNDICE K - Topologia com 20 switches: configurações nos componentes de rede no rack OpenFlow InstaGENI da Universidade de UtahDDC.

Node #1

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s9	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s9.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39235 ssh ligiapre@pc15.utahddc.geniracks.net -p 39235				
Interfaces		**MAC**		**Layer 3**	
interface-18	pc15:lo0	0227ce2aeb15		ipv4: 10.10.1.18	
interface-44	pc15:eth1	0255b58606fd		ipv4: 10.10.1.44	
interface-52	pc15:eth1	02be95e2d996		ipv4: 10.10.1.52	
interface-81	pc15:lo0	0274d911a19d		ipv4: 10.10.1.81	
interface-83	pc15:lo0	028afee65cfe		ipv4: 10.10.1.83	

Node #2

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s1	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s1.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39229 ssh ligiapre@pc10.utahddc.geniracks.net -p 39229				
Interfaces		**MAC**		**Layer 3**	
interface-2	pc10:eth3	02837418f186		ipv4: 10.10.1.2	
interface-41	pc10:lo0	0207d60c0796		ipv4: 10.10.1.41	
interface-43	pc10:eth3	0235f279f12c		ipv4: 10.10.1.43	
interface-45	pc10:lo0	02203cfb4302		ipv4: 10.10.1.45	
interface-47	pc10:lo0	027372e078d2		ipv4: 10.10.1.47	

Node #3

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s2	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s2.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39233 ssh ligiapre@pc10.utahddc.geniracks.net -p 39233				
Interfaces		**MAC**		**Layer 3**	
interface-4	pc10:eth3	02ca2c176600		ipv4: 10.10.1.4	
interface-49	pc10:lo0	02af4ba65806		ipv4: 10.10.1.49	
interface-51	pc10:eth3	02c599226d7b		ipv4: 10.10.1.51	
interface-53	pc10:lo0	02bedc5c196c		ipv4: 10.10.1.53	
interface-55	pc10:lo0	025a73c5dc17		ipv4: 10.10.1.55	

Node #4

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s3	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s3.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39232 ssh ligiapre@pc15.utahddc.geniracks.net -p 39232				
Interfaces		**MAC**		**Layer 3**	
interface-6	pc15:lo0	02322fcbb5b0		ipv4: 10.10.1.6	
interface-57	pc15:lo0	02fbfdc14267		ipv4: 10.10.1.57	
interface-59	pc15:lo0	02c19816eb32		ipv4: 10.10.1.59	
interface-61	pc15:lo0	02d5482a83af		ipv4: 10.10.1.61	
interface-63	pc15:lo0	026dcba2d820		ipv4: 10.10.1.63	

Node #5

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s4	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s4.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39233 ssh ligiapre@pc15.utahddc.geniracks.net -p 39233				

Interfaces		MAC	Layer 3
interface-8	pc15:lo0	029e1c1d8c3a	ipv4: 10.10.1.8
interface-65	pc15:lo0	020b455b02e6	ipv4: 10.10.1.65
interface-67	pc15:lo0	029c9dcb0675	ipv4: 10.10.1.67
interface-69	pc15:lo0	029fb39c0b52	ipv4: 10.10.1.69
interface-71	pc15:lo0	02f46ba26a2b	ipv4: 10.10.1.71

Node #6

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s5	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s5.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39235 ssh ligiapre@pc10.utahddc.geniracks.net -p 39235				

Interfaces		MAC	Layer 3
interface-10	pc10:eth3	026cc15d0a63	ipv4: 10.10.1.10
interface-42	pc10:lo0	02f0cf3f674b	ipv4: 10.10.1.42
interface-50	pc10:lo0	026445b7a505	ipv4: 10.10.1.50
interface-75	pc10:eth3	028f217ba0c3	ipv4: 10.10.1.75
interface-73	pc10:eth3	02693bbacdac	ipv4: 10.10.1.73

Node #7

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s6	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s6.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39234 ssh ligiapre@pc15.utahddc.geniracks.net -p 39234				

Interfaces		MAC	Layer 3
interface-12	pc15:lo0	026ef1b0993a	ipv4: 10.10.1.12
interface-58	pc15:lo0	0217651db656	ipv4: 10.10.1.58
interface-66	pc15:lo0	021f85eac0b1	ipv4: 10.10.1.66
interface-77	pc15:eth1	026bf1b2dbfa	ipv4: 10.10.1.77
interface-79	pc15:eth1	02caccab2328	ipv4: 10.10.1.79

Node #8

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s7	pc5	2016-12-31T23:59:59.000Z	emulab-xen	s7.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc5.utahddc.geniracks.net -p 39227 ssh ligiapre@pc5.utahddc.geniracks.net -p 39227				

Interfaces		MAC	Layer 3
interface-14	pc5:eth3	023a564b1b0e	ipv4: 10.10.1.14
interface-78	pc5:eth3	02dc4d01cdd5	ipv4: 10.10.1.78
interface-74	pc5:eth3	02af2937d262	ipv4: 10.10.1.74
interface-105	pc5:lo0	02283c0cfe7b	ipv4: 10.10.1.105
interface-107	pc5:eth3	0291a4a88d36	ipv4: 10.10.1.107

Node #9

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s8	pc4	2016-12-31T23:59:59.000Z	emulab-xen	s8.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc4.utahddc.geniracks.net -p 39227 ssh ligiapre@pc4.utahddc.geniracks.net -p 39227				
Interfaces		**MAC**		**Layer 3**	
interface-16	pc4:eth1	02716be21364		ipv4: 10.10.1.16	
interface-76	pc4:eth1	0274a5e7f817		ipv4: 10.10.1.76	
interface-80	pc4:eth1	0220fdd72cfb		ipv4: 10.10.1.80	
interface-109	pc4:lo0	0272d4444ca3		ipv4: 10.10.1.109	
interface-111	pc4:eth1	02baad2c7587		ipv4: 10.10.1.111	

Node #10

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s10	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s10.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39227 ssh ligiapre@pc15.utahddc.geniracks.net -p 39227				
Interfaces		**MAC**		**Layer 3**	
interface-20	pc15:lo0	0294acacaddd		ipv4: 10.10.1.20	
interface-60	pc15:lo0	027d2a1c40ce		ipv4: 10.10.1.60	
interface-68	pc15:lo0	023a14d2715b		ipv4: 10.10.1.68	
interface-85	pc15:lo0	021d3c94c596		ipv4: 10.10.1.85	
interface-87	pc15:lo0	025a556ff2aa		ipv4: 10.10.1.87	

Node #11

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h1	pc5	2016-12-31T23:59:59.000Z	emulab-xen	h1.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc5.utahddc.geniracks.net -p 39226 ssh ligiapre@pc5.utahddc.geniracks.net -p 39226				
Interfaces		**MAC**		**Layer 3**	
interface-106	pc5:lo0	02a9825fd31f		ipv4: 10.10.1.106	

Node #12

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h2	pc14	2016-12-31T23:59:59.000Z	emulab-xen	h2.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc14.utahddc.geniracks.net -p 39226 ssh ligiapre@pc14.utahddc.geniracks.net -p 39226				
Interfaces		**MAC**		**Layer 3**	
interface-108	pc14:eth3	0256de825f8a		ipv4: 10.10.1.108	

Node #13

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h3	pc4	2016-12-31T23:59:59.000Z	emulab-xen	h3.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc4.utahddc.geniracks.net -p 39226 ssh ligiapre@pc4.utahddc.geniracks.net -p 39226				
Interfaces		**MAC**		**Layer 3**	
interface-110	pc4:lo0	02f873c1f73b		ipv4: 10.10.1.110	

Node #14

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h4	pc12	2016-12-31T23:59:59.000Z	emulab-xen	h4.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc12.utahddc.geniracks.net -p 39226 ssh ligiapre@pc12.utahddc.geniracks.net -p 39226				
Interfaces		MAC		Layer 3	
interface-112	pc12:eth2	0221c4222fe7		ipv4: 10.10.1.112	

Node #15

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h5	pc7	2016-12-31T23:59:59.000Z	emulab-xen	h5.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc7.utahddc.geniracks.net -p 39226 ssh ligiapre@pc7.utahddc.geniracks.net -p 39226				
Interfaces		MAC		Layer 3	
interface-114	pc7:eth3	02c502b86f15		ipv4: 10.10.1.114	

Node #16

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h6	pc3	2016-12-31T23:59:59.000Z	emulab-xen	h6.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc3.utahddc.geniracks.net -p 39226 ssh ligiapre@pc3.utahddc.geniracks.net -p 39226				
Interfaces		MAC		Layer 3	
interface-116	pc3:eth2	02298629f317		ipv4: 10.10.1.116	

Node #17

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h7	pc1	2016-12-31T23:59:59.000Z	emulab-xen	h7.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc1.utahddc.geniracks.net -p 39226 ssh ligiapre@pc1.utahddc.geniracks.net -p 39226				
Interfaces		MAC		Layer 3	
interface-118	pc1:eth2	020642e30930		ipv4: 10.10.1.118	

Node #18

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h8	pc6	2016-12-31T23:59:59.000Z	emulab-xen	h8.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc6.utahddc.geniracks.net -p 39226 ssh ligiapre@pc6.utahddc.geniracks.net -p 39226				
Interfaces		MAC		Layer 3	
interface-120	pc6:eth1	028ab3428072		ipv4: 10.10.1.120	

Node #19

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	controller	pc15	2016-12-31T23:59:59.000Z	emulab-xen	controller.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39226 ssh ligiapre@pc15.utahddc.geniracks.net -p 39226				

Interfaces		MAC	Layer 3
interface-1	pc15:eth1	02cc9ac4b019	ipv4: 10.10.1.1
interface-3	pc15:eth3	026e7e7bdff1	ipv4: 10.10.1.3
interface-5	pc15:lo0	02011c5a9c7e	ipv4: 10.10.1.5
interface-7	pc15:lo0	02fe159c8073	ipv4: 10.10.1.7
interface-9	pc15:eth3	021708259dd3	ipv4: 10.10.1.9
interface-11	pc15:lo0	02fc557c751d	ipv4: 10.10.1.11
interface-13	pc15:eth3	02c5af077f77	ipv4: 10.10.1.13
interface-15	pc15:eth3	022f8a9b5681	ipv4: 10.10.1.15
interface-17	pc15:lo0	024ce4e5a48c	ipv4: 10.10.1.17
interface-19	pc15:lo0	024913af0062	ipv4: 10.10.1.19
interface-21	pc15:lo0	020bbb5acaae	ipv4: 10.10.1.21
interface-23	pc15:lo0	027391dde3a5	ipv4: 10.10.1.23
interface-25	pc15:eth1	029eec7a15ce	ipv4: 10.10.1.25
interface-27	pc15:lo0	028b63f73d9c	ipv4: 10.10.1.27
interface-29	pc15:eth1	024ff392bd3a	ipv4: 10.10.1.29
interface-31	pc15:eth3	021a9c001511	ipv4: 10.10.1.31
interface-33	pc15:eth3	02280c2db5f1	ipv4: 10.10.1.33
interface-35	pc15:lo0	02e0b0337c4e	ipv4: 10.10.1.35
interface-37	pc15:eth3	0299be2e5cd6	ipv4: 10.10.1.37
interface-39	pc15:eth3	027e47027b01	ipv4: 10.10.1.39

Node #20

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h9	pc10	2016-12-31T23:59:59.000Z	emulab-xen	h9.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39228 ssh ligiapre@pc10.utahddc.geniracks.net -p 39228				

Interfaces		MAC	Layer 3
interface-122	pc10:lo0	024368b7ca45	ipv4: 10.10.1.122

Node #21

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h10	pc10	2016-12-31T23:59:59.000Z	emulab-xen	h10.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39226 ssh ligiapre@pc10.utahddc.geniracks.net -p 39226				

Interfaces		MAC	Layer 3
interface-124	pc10:lo0	02443966cdd6	ipv4: 10.10.1.124

Node #22

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h11	pc11	2016-12-31T23:59:59.000Z	emulab-xen	h11.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc11.utahddc.geniracks.net -p 39226 ssh ligiapre@pc11.utahddc.geniracks.net -p 39226				

Interfaces		MAC	Layer 3
interface-126	pc11:lo0	02d91777056c	ipv4: 10.10.1.126

Node #23

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h12	pc11	2016-12-31T23:59:59.000Z	emulab-xen	h12.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc11.utahddc.geniracks.net -p 39227 ssh ligiapre@pc11.utahddc.geniracks.net -p 39227				
Interfaces		**MAC**		**Layer 3**	
interface-128	pc11:lo0	025ec94633d9		ipv4: 10.10.1.128	

Node #24

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h13	pc8	2016-12-31T23:59:59.000Z	emulab-xen	h13.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc8.utahddc.geniracks.net -p 39226 ssh ligiapre@pc8.utahddc.geniracks.net -p 39226				
Interfaces		**MAC**		**Layer 3**	
interface-130	pc8:lo0	02fbfc4a3a72		ipv4: 10.10.1.130	

Node #25

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h14	pc13	2016-12-31T23:59:59.000Z	emulab-xen	h14.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc13.utahddc.geniracks.net -p 39226 ssh ligiapre@pc13.utahddc.geniracks.net -p 39226				
Interfaces		**MAC**		**Layer 3**	
interface-132	pc13:eth3	02f2f312cbbf		ipv4: 10.10.1.132	

Node #26

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h16	pc10	2016-12-31T23:59:59.000Z	emulab-xen	h16.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39227 ssh ligiapre@pc10.utahddc.geniracks.net -p 39227				
Interfaces		**MAC**		**Layer 3**	
interface-136	pc10:lo0	02146987729c		ipv4: 10.10.1.136	

Node #27

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	h15	pc2	2016-12-31T23:59:59.000Z	emulab-xen	h15.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc2.utahddc.geniracks.net -p 39226 ssh ligiapre@pc2.utahddc.geniracks.net -p 39226				
Interfaces		**MAC**		**Layer 3**	
interface-134	pc2:eth3	02659c7ab872		ipv4: 10.10.1.134	

Node #28

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s11	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s11.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39228 ssh ligiapre@pc15.utahddc.geniracks.net -p 39228				
Interfaces		**MAC**		**Layer 3**	
interface-22	pc15:lo0	02428ef52ee6		ipv4: 10.10.1.22	
interface-82	pc15:lo0	02770337967f		ipv4: 10.10.1.82	
interface-86	pc15:lo0	02fd46e44841		ipv4: 10.10.1.86	
interface-113	pc15:eth1	0255af752812		ipv4: 10.10.1.113	
interface-115	pc15:eth1	02d3093661d6		ipv4: 10.10.1.115	

Node #29

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s12	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s12.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39229 ssh ligiapre@pc15.utahddc.geniracks.net -p 39229				

Interfaces		MAC	Layer 3
interface-24	pc15:lo0	02f2c68f3c9f	ipv4: 10.10.1.24
interface-84	pc15:lo0	023f27340285	ipv4: 10.10.1.84
interface-88	pc15:lo0	024cc7447865	ipv4: 10.10.1.88
interface-117	pc15:eth1	026c1fe12ba3	ipv4: 10.10.1.117
interface-119	pc15:eth1	02003fac53e0	ipv4: 10.10.1.119

Node #30

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s13	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s13.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39230 ssh ligiapre@pc10.utahddc.geniracks.net -p 39230				

Interfaces		MAC	Layer 3
interface-26	pc10:eth3	02f6728a4d3d	ipv4: 10.10.1.26
interface-46	pc10:lo0	02543f86904d	ipv4: 10.10.1.46
interface-54	pc10:lo0	029de7c35ca1	ipv4: 10.10.1.54
interface-89	pc10:lo0	02e68cc49155	ipv4: 10.10.1.89
interface-91	pc10:eth3	020991d3e19d	ipv4: 10.10.1.91

Node #31

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s14	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s14.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39230 ssh ligiapre@pc15.utahddc.geniracks.net -p 39230				

Interfaces		MAC	Layer 3
interface-28	pc15:lo0	022e395dc572	ipv4: 10.10.1.28
interface-62	pc15:lo0	02ccd3cfec9c	ipv4: 10.10.1.62
interface-70	pc15:lo0	020c8e0f9fe0	ipv4: 10.10.1.70
interface-93	pc15:eth1	02fbb3cc4b14	ipv4: 10.10.1.93
interface-95	pc15:eth1	02451359c876	ipv4: 10.10.1.95

Node #32

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s15	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s15.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39231 ssh ligiapre@pc10.utahddc.geniracks.net -p 39231				

Interfaces		MAC	Layer 3
interface-30	pc10:eth3	02936d335a71	ipv4: 10.10.1.30
interface-90	pc10:lo0	02748f8d7cd9	ipv4: 10.10.1.90
interface-94	pc10:eth3	020111193f11	ipv4: 10.10.1.94
interface-121	pc10:lo0	028be55d2693	ipv4: 10.10.1.121
interface-123	pc10:lo0	021d685f4a4e	ipv4: 10.10.1.123

Node #33

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s16	pc11	2016-12-31T23:59:59.000Z	emulab-xen	s16.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc11.utahddc.geniracks.net -p 39228 ssh ligiapre@pc11.utahddc.geniracks.net -p 39228				

Interfaces		MAC	Layer 3
interface-32	pc11:eth1	027a30bcce89	ipv4: 10.10.1.32
interface-92	pc11:eth1	02344c2d0873	ipv4: 10.10.1.92
interface-96	pc11:eth1	0258cb1adfb6	ipv4: 10.10.1.96
interface-125	pc11:lo0	020c319be089	ipv4: 10.10.1.125
interface-127	pc11:lo0	0260fdfaebcb	ipv4: 10.10.1.127

Node #34

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s17	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s17.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39232 ssh ligiapre@pc10.utahddc.geniracks.net -p 39232				

Interfaces		MAC	Layer 3
interface-34	pc10:eth3	02b7268ead58	ipv4: 10.10.1.34
interface-48	pc10:lo0	02ca824799dd	ipv4: 10.10.1.48
interface-56	pc10:lo0	0209fd777941	ipv4: 10.10.1.56
interface-97	pc10:eth3	02c49701c7e2	ipv4: 10.10.1.97
interface-99	pc10:lo0	027796ff0a68	ipv4: 10.10.1.99

Node #35

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s18	pc15	2016-12-31T23:59:59.000Z	emulab-xen	s18.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc15.utahddc.geniracks.net -p 39231 ssh ligiapre@pc15.utahddc.geniracks.net -p 39231				

Interfaces		MAC	Layer 3
interface-36	pc15:lo0	02b09bffe27d	ipv4: 10.10.1.36
interface-64	pc15:lo0	027b1665979a	ipv4: 10.10.1.64
interface-72	pc15:lo0	023f8c8b078e	ipv4: 10.10.1.72
interface-101	pc15:eth1	02a1005edae7	ipv4: 10.10.1.101
interface-103	pc15:eth1	02a8fdde64b4	ipv4: 10.10.1.103

Node #36

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s19	pc8	2016-12-31T23:59:59.000Z	emulab-xen	s19.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc8.utahddc.geniracks.net -p 39227 ssh ligiapre@pc8.utahddc.geniracks.net -p 39227				

Interfaces		MAC	Layer 3
interface-38	pc8:eth2	02a3b480d5a4	ipv4: 10.10.1.38
interface-98	pc8:eth2	0282c6d0e4ec	ipv4: 10.10.1.98
interface-102	pc8:eth2	024cf7691eb0	ipv4: 10.10.1.102
interface-129	pc8:lo0	02aa460aad78	ipv4: 10.10.1.129
interface-131	pc8:eth2	02fb30cac0ba	ipv4: 10.10.1.131

Node #37

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	s20	pc10	2016-12-31T23:59:59.000Z	emulab-xen	s20.FatTree16.ch-geni-net.utahddc.geniracks.net
Login	ssh chris@pc10.utahddc.geniracks.net -p 39234 ssh ligiapre@pc10.utahddc.geniracks.net -p 39234				

Interfaces		MAC	Layer 3
interface-40	pc10:eth3	029a4b6830cd	ipv4: 10.10.1.40
interface-100	pc10:lo0	0265ba2b6f2b	ipv4: 10.10.1.100
interface-104	pc10:eth3	029986a7cb0e	ipv4: 10.10.1.104
interface-133	pc10:eth3	02490f7d330f	ipv4: 10.10.1.133
interface-135	pc10:lo0	02d50f72e70f	ipv4: 10.10.1.135