UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Compartilhamento de Banda em Redes Definidas por Software

Dissertação de Mestrado

Glauco Luiz Rezende de Carvalho

São Cristóvão – Sergipe

2018

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Glauco Luiz Rezende de Carvalho

Compartilhamento de Banda em Redes Definidas por Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Ricardo José P. de Britto Salgueiro
Coorientador(a): Profa. Dra. Edilayne Meneses Salgueiro

São Cristóvão – Sergipe

2018

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL UNIVERSIDADE FEDERAL DE SERGIPE

Carvalho, Glauco Luiz Rezende de
C331c Compartilhamento de banda em redes definidas por software / Glauco Luiz Rezende de Carvalho ; orientador Ricardo José P. de Brito Salgueiro. – São Cristóvão, 2017.
77 f. : il.

Dissertação (mestrado em Ciências da computação)– Universidade Federal de Sergipe, 2017.

1. Programas de computador. 2. Redes de computadores. 3. Engenharia de software. 4. Software. 5. Sistemas de comunicação em banda larga. I. Salgueiro, Ricardo José P. de Brito, orient. II. Título.

CDU 004.4

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÂO E PESQUISA
NÚCLEO DE PÓS-GRADUAÇÂO EM CIÊNCIA DA COMPUTAÇÃO

Relatório de defesa pública do (a) Senhor (a) **GLAUCO LUIZ REZENDE DE CARVALHO** no Programa de Ciência da Computação (PROCC) da UFS.

Aos 31 dias de junho de 2017, realizou-se a **Defesa de Mestrado** do trabalho intitulado **"Compartilhamento de Banda em Redes Definidas por Softwares"** sob orientação do Profº. Drº. **Ricardo José Paiva de BrittoSalgueiro.**

Depois de declarada aberta a sessão, o Presidente da Banca passou inicialmente a palavra ao candidato para exposição e a seguir aos examinadores para as devidas arguições que se desenvolveram nos termos regimentais. Em seguida, a comissão julgadora proclamou o resultado:

Nome Banca Examinadora	Instituição	Nota (de 0 a 10)	Assinatura
Ricardo José Paiva de Britto Salgueiro	UFS	Aprovado	
Edward David Moreno Ordeonez	UFS	Aprovado	
Edilayne Meneses Salgueiro	UFPE	Aprovado	
Willian Ferreira Giozza	UnB	Aprovado	
	Média =	Aprovado	

Dessa maneira o Resultado Final é: X **APROVADO** ou **REPROVADO**

Parecer da Banca Examinadora *

- Obs: Se o candidato for reprovado, o preenchimento do parecer é obrigatório.

São Cristóvão/SE

Assinatura do Orientador:

Assinatura do Aluno:

Dedico esta dissertação às minhas filhas, minha vida, presente de Deus! Obrigado meus amores por vocês tornarem minha vida melhor e mais feliz. Saibam que vocês são a razão pela qual existo. Vocês são toda minha razão. Obrigado. Amo vocês!

Agradecimentos

Em primeiro lugar gostaria de agradeço a Deus, por me dar força e coragem em todos os momentos desses dois anos e meio de caminhada. Tenho a certeza de que Suas ações, principalmente nos detalhes, foram decisivas nas decisões que tomei nos últimos anos. Que Ele continue me abençoando, nos bons e maus momentos, para sempre!

Agradeço às minhas filhas que tanto amo, Letícia e Luiza pela compreensão nos momentos de ausência, aos meus pais, Lígia e Geraldo (*in memorian*) e ao meu avô Luiz (*in memorian*), pois sem vocês, nada seria possível. As minhas conquistas são, na verdade, suas conquistas.

Agradeço aos meus irmãos Grace e Gustavo pelo incentivo dado em todos os momentos da minha trajetória acadêmica. Tenho certeza de que coisas boas estão reservadas para nós. Amo vocês.

Um agradecimento especial ao meu orientador, Prof. Dr. Ricardo José P. de Britto Salgueiro, à minha coorientadora Profa. Dra. Edilayne Meneses Salgueiro e ao amigo Andeson Morais, pelo apoio incondicional durante a realização desta dissertação.

Agradeço aos amigos e colegas de mestrado que estiveram ao meu lado nessa caminhada. Obrigado por tornarem muito mais fácil e mais empolgante essa longa jornada.

Agradeço à UFS e a PROCC por possibilitarem essa formação. Obrigado por me enriquecerem pessoalmente e profissionalmente.

A todos, um obrigado!

Resumo

O paradigma de Redes Definidas por *Software* (SDN) vem impondo mudanças significativas na forma de gerenciar e operar redes de computadores através da sua principal ideia, a separação dos planos de dados e de controle. O protocolo OpenFlow suporta e implementa este conceito e devido às diversas vantagens de menor custo de operação e propiciar maior facilidade de adaptação a projetos computacionais já existentes, é facilmente encontrado em diversos equipamentos de rede comercializados. Com a adoção do protocolo OpenFlow e do paradigma SDN, a inovação e evolução da rede são relativamente mais favorecidas quando comparadas com o modelo tradicional. Dessa forma, serviços comuns de rede podem ser evoluídos, tornando-os cada vez mais flexíveis. Neste sentido, devido à padronização e adaptabilidade propostas pelas SDNs, e das limitações na arquitetura atual das redes IPs, esta dissertação propõe uma técnica que utiliza uma rede baseada na arquitetura SDN e o protocolo OpenFlow para a criação de um serviço para compartilhamento de banda, utilizando o controlador Floodlight. Emulações em ambiente virtual Mininet de experimentação foram desenvolvidas para validação das implementações realizadas. Para a análise de desempenho, foram estudadas métricas como largura de banda, número de pacotes enviados, recebidos e descartados. Os resultados demonstraram a possibilidade de alcançar com eficiência o compartilhamento de banda em redes SDN.

Palavras-chave: rede, software, alocação, compartilhamento, largura de banda, openflow, floodlight, mininet.

Abstract

The paradigm of Software Defined Networks (SDN) has imposed significant changes in the way to manage and operate computer networks through its main idea, the separation of data and control plans. The OpenFlow protocol supports and implements this concept and, due to the many advantages of lower cost of operation and greater ease of adaptation to existing computational projects, is easily found in several commercially available network equipment. With the adoption of the OpenFlow protocol and the SDN paradigm, network innovation and evolution are relatively more favored when compared to the traditional model. In this way, common network services can be evolved, making them increasingly flexible. This dissertation proposes a technique that uses a network based on the SDN architecture and the OpenFlow protocol for the creation of a service for bandwidth sharing, Using the Floodlight controller. Emulations in Mininet virtual environment of experimentation were developed to validate the implemented implementations. For the performance analysis, we studied metrics such as bandwidth, number of packets sent, received and discarded. The results demonstrated the possibility of efficiently achieving bandwidth sharing in SDN networks.

Keywords: network, software, allocation, sharing, bandwidth, openflow, floodlight, mininet.

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Lista de abreviaturas e siglas

ACK	Acknowledgement
API	Application Programming Interface
ATM	Asynchronus Transfer Mode
BGP	Border Gateway Protocol
BSS	Bandwidth Sharing Service on Software-Defined Networks
CAB	CAching in Buckets
CORBA	Common Object Request Broker Architecture
CT	Class Type
DER	Relationship Entity Diagram
DiffServ	Differentiated Services
DS-TE	DiffServ-Aware Traffic Engineering
DSCP	Differentiated Service Code Point
EDGE	Enhanced Data rates for GSM Evolution
FIN	Finally
GSM	Groupe Special Mobile
GSMP	General Switch Management Protocol
HSPA	High Speed Downlink Packet Access
HTTP	Hypertext Transfer Protocol
ICMPv6	Internet Control Message Protocol version 6
ID	Identity
IP	Internet Protocol
IPv6	Internet Protocol version 6
ISIS	Intermediate System to Intermediate System
LACP	Link Aggregation Control Protocol

LDP	Label Distribution Protocol
LTE	Long Term Evolution
MAC	Media Access Control
MAM	Maximum Allocation Model
MPLS	Multiprotocol Label Switching
NGN	Next Generation Networks
ONF	Open Networking Foundation
OSPF	Open Shortest Path First
OvS	Open vSwitch
OVSDB	Open vSwitch Database Management Protocol
PCE	Path Computation Element
PPP	Point-to-Point Protocol
QoS	Quality of Service
RDM	Russian Doll Model
REST	Representational State Transfer
RPC	Remote Procedure Calling Protocol
RSVP	Resource Reservation Protocol
SDN	Software-Defined Networks
SLA	Service-Level Agreements
SOAP	Simple Object Access Protocol
SRAM	Static Random Access Memory
SYN	Synchronize
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TI	Information Technology
TLS	Transport Layer Security

TS	Type of Service
TTL	Time to Live
UDP	User Datagram Protocol
UFS	Universidade Federal de Sergipe
UMTS	Universal Mobile Telecommunication System
VLAN	Virtual Local Area Network
VM	Virtual Machine
VoIP	Voice over Internet Protocol

Sumário

1 Introdução

Este trabalho apresenta um mecanismo de compartilhamento de banda em Redes Definidas por *Software* (SDN). Este capítulo introduz o contexto onde se enquadram os conceitos de Compartilhamento de Banda e Redes SDN utilizados. Em seguida, apresenta o objeto de estudo, a estratégia adotada e a organização deste trabalho.

As redes corporativas atuais são compostas de diversos equipamentos e aplicações. O aumento da escala de uma rede corporativa faz com que as interações entre as aplicações se tornem mais complexas e envolvam diversos elementos de *hardware* e *software*, como: *switches* e roteadores. Estes equipamentos, quando mais modernos são dotados normalmente de um plano de dados e um plano de controle. O plano de dados é o elemento responsável por todo o encaminhamento do tráfego de rede entre as portas do equipamento e o plano de controle é o responsável por definir como o encaminhamento deve ser feito no plano de dados. Sendo assim no modelo atual, qualquer definição e modificação do plano de controle só pode ser realizada pelo fabricante do equipamento. Isto, consequentemente, obriga os administradores e usuários a ficarem dependentes de seus fornecedores no que se refere a novas funcionalidades implementadas e disponibilizadas nos equipamentos de rede. Essa arquitetura tem sido fortemente criticada por ser de difícil alteração e evolução, sendo inclusive rotulada de ossificada por alguns pesquisadores.

As redes de nova geração NGN - *Next Generation Networks* utilizam uma única infraestrutura de redes para o transporte de voz, dados e multimídia, também denominadas como redes multisserviços. A convergência de redes é a palavra-chave para as telecomunicações do futuro. Outro aspecto importante em redes multisserviços é a alocação de recursos limitados em um cenário onde as demandas são conflitantes. Alocação de recursos é o processo realizado por alguns elementos da rede para atender à demanda de aplicações (KIM, 2014). Os mecanismos de alocação devem ser capazes de atribuir de maneira eficiente os recursos disponíveis em uma rede entre as diferentes aplicações que competem por eles, maximizando assim o uso dos

recursos. Neste contexto, é importante ressaltar que a capacidade de transmissão da rede ou simplesmente largura de banda, diferentemente da água ou combustível, é um recurso que não pode ser armazenado para uso posterior e, portanto, seu excesso é desperdiçado (SALGUEIRO, 2009).

Nesse contexto, segundo Nunes et al. (2014b), as redes definidas por software, (*Software Defined Network* - SDN), surgem como a tecnologia chave para promover a inovação das redes multis-serviços. A ideia do conceito SDN é proporcionar a dissociação entre o plano de dados do plano de controle. Desta forma, é possível usufruir de várias vantagens, como dispor de um controle maior da rede, de monitoramento eficiente e de um controlador com gerenciamento local ou remoto de todos os elementos da rede. Esse gerenciamento é efetuado através de aplicações sistêmicas, da mesma forma como é feito com *softwares* para *desktop*, daí a denominação conferida a este novo conceito que é recente. Atualmente, o foco principal das pesquisas realizadas com esse tema está relacionado em como utilizar esse novo conceito para resolver problemas das redes de computadores tradicionais. Essa mesma abordagem foi amplamente discutida em (PATEL et al., 2014; FERGUSON et al., 2012; JO; LEE; KIM, 2014; LUN; GRACE, 2015).

Segundo Lara, Kolasani e Ramamurthy (2014) e Nunes et al. (2014b), o protocolo *OpenFlow* foi concebido para trabalhar com a arquitetura SDN, é reconhecido como o padrão nesse segmento e é um dos protocolos de comunicação utilizados entre os elementos situados na camada de controle e os elementos na camada de infraestrutura. De forma geral, essa comunicação ocorre entre o controlador da rede e um equipamento que entenda o protocolo *OpenFlow*, normalmente chamado de *OpenFlow Switch* (NAKAGAWA; HYOUDOU; SHIMIZU, 2012). Este possui uma interface de comunicação que possibilita a comunicação com o controlador e é através dela que o controlador pode manipular as informações contidas no *switch*.

Neste trabalho, é utilizada a arquitetura de SDN para desenvolver uma técnica que possibilita a utilização do protocolo OpenFlow no desenvolvimento de um mecanismo para compartilhamento de banda em um controlador SDN. O árbitro possui capacidade gerencial para definir a quem e quando deverá ou não realizar empréstimos de banda. Desta forma, o árbitro promoverá de forma automática e eficiente a utilização da largura de banda total disponível em SDN.

1.1 Problemática

As redes de computadores oferecem a cada dia uma variedade imensa de serviços como serviços de voz (*Voice over IP* - VoIP), interconexões privadas, televisão digital entre outros. Por estas razões, estas redes transportam uma grande variedade de serviços com requisitos de QoS distintos, os quais adicionados ao grande volume de tráfego e à dimensão das redes tornam o gerenciamento destas uma tarefa árdua e complexa.

A complexidade arquitetural é outro importante item nas redes multisserviços que possuem mecanismos automáticos de controle geograficamente distribuídos, tais como protocolos de roteamento e sinalização, a exemplo do (*Open Shortest Path First* - OSPF), (*Border Gateway Protocol* - BGP), (*Intermediate System to Intermediate System* - ISIS); e (*Resource Reservation Protocol* - RSVP) e (*Label Distribution Protocol* - LDP) respectivamente. Estes elementos de controle proveem arquiteturas complexas, capazes de automaticamente recalcular suas topologias caso haja falha em um dos equipamentos. Contudo, tais mecanismos levam a equipamentos complexos, responsáveis por construir e manter as tabelas de roteamento nos mais diversos encaminhamentos de tráfego.

Os mecanismos de compartilhamento de banda em redes, oferecem benefícios e são amplamente discutidos em diversos trabalhos na literatura, a exemplo de Awduche et al. (1999), Salgueiro (2009), Shnayder et al. (2014), Wagner (2014), Hamouda, Ben Chaabane e Tabbane (2015), Mohan, Divakaran e Gurusamy (2016), em que os autores apresentam e discutem estratégias e métodos para compartilhamento de banda, a fim de maximizar o uso destas redes, simplificando o gerenciamento e promovendo um compartilhamento de banda eficiente.

Como características adicionais, as redes multiserviços são caras em termos de infraestrutura e operação, possuindo administradores dedicados e especializados em diferentes tipos de tecnologias. Em sua grande maioria, os equipamentos utilizados pertencem a um pequeno conjunto de fabricantes, que encapsulam a inteligência da rede em seus equipamentos, que de certa forma tornam as inovações mais lentas e restritas aos anseios dos detentores das tecnologias embarcadas.

1.2 Hipótese

Um controlador SDN pode prover um mecanismo de arbitragem para compartilhamento e empréstimo de banda de forma eficiente em uma rede multisserviço.

1.3 Objetivos

Este trabalho tem como objetivo prover um serviço de compartilhamento de banda em redes definidas por *software*, utilizando um modelo de barganha. Assim, como objetivo específico tem-se definidos:

- Definir uma arquitetura de serviço que possibilite o compartilhamento de banda em redes definidas por *software*;
- Implementar um mecanismo de barganha através de algoritmos;
- Implementar um *software* protótipo que suporte o serviço proposto;

1.4 Organização do Trabalho

Esta dissertação está divida em seis capítulos, onde o segundo capítulo é apresentada a fundamentação teórica no contexto de uma revisão literária sobre redes definidas por *softwares*. O terceiro capítulo apresenta os trabalhos relacionados sobre modelos de alocação de banda, redes definidas por *softwares* e o compartilhamento de banda em SDN. É apresentada no quarto capítulo a solução proposta para o desenvolvimento do compartilhamento de banda em redes definidas por *softwares* para esta dissertação, incluindo os recursos lógicos e físicos utilizados. O quinto capítulo um estudo de caso, bem como as análises dos resultados. Por fim, o sexto capítulo apresenta as conclusões desta dissertação, suas contribuições e as sugestões dos trabalhos futuros.

2

Redes Definidas por Software

Este capítulo apresenta características fundamentais do paradigma de Redes Definidas por *Software*. É abordado inicialmente na seção 2.1 pontos chave deste paradigma, além de alguns conceitos e características essenciais. A seção 2.2, apresenta o desenvolvimento do paradigma SDN, desde a sua concepção à sua proposição, passando pela formulação de seus conceitos fundamentais e terminando com a definição atual do paradigma. A seção 2.3 aborda as características essenciais dos elementos que compõem essa arquitetura. Já a seção 2.4 com profundidade, uma vez que o mesmo é peça chave para a implementação de redes SDN. Por fim, é abordado na seção 2.5, exemplos de aplicações que podem beneficiar-se com a utilização desse paradigma.

2.1 Definição

De maneira sucinta, Redes Definidas por *Software* (SDN) são um paradigma de redes de computadores que define a realização do encaminhamento de pacotes de uma maneira diferente da tradicional. A implementação do paradigma consiste na inserção de um novo elemento na rede - o controlador – que passa a ser o responsável por definir as ações que devem ser tomadas pelos elementos comutadores (*switches* ou roteadores) no encaminhamento de pacotes (MACEDO et al., 2015). Essa proposta de separar a entidade controladora e tomadora de decisões (plano de controle) da que executa o que foi determinado (plano de dados) é justamente a base sobre a qual está assentada a formulação do paradigma. Assim, essa nova ideia de inserir controle sobre as operações da rede oferece um contraponto ao modelo tradicional de repasse de dados, no qual o controle e a operação de encaminhamento são executados pelo equipamento de comutação da rede. No entanto, a ideia por trás do paradigma SDN é muito mais profunda e significativa do que a simples inserção de um elemento de controle na rede. As definições a seguir trata alguns aspectos contidos na formulação do paradigma de uma forma mais detalhada.

O termo SDN, em si, foi criado recentemente, contudo todo o conceito por trás da sigla SDN vem sendo desenvolvido desde meados da década de 1990 (ZILBERMAN et al., 2015). Portanto, múltiplas definições do que vem a ser de fato SDN têm sido elaboradas ao longo dos últimos anos. Uma das definições amplamente utilizadas é justamente a da *Open Networking Foundation* (ONF), a organização criada para a promoção, adoção e implementação de SDN na indústria, citada anteriormente. Segundo essa definição, SDN é uma arquitetura emergente, dinâmica, gerenciável, de ótimo custo benefício, adaptável, ideal para a natureza dinâmica das aplicações de hoje, e que deve, principalmente, ser capaz de separar o controle da rede e as funções de encaminhamento, permitindo que o controle da rede se torne diretamente programável e que a infraestrutura subjacente seja abstraída para aplicações e serviços de rede (NUNES et al., 2014b; MCKEOWN et al., 2008).

De forma mais específica, o ponto chave da definição anterior é o mesmo dos trabalhos de Nunes et al. (2014b); Zilberman et al. (2015); Sezer et al. (2013); Kreutz et al. (2015); Feamster, Rexford e Zegura (2014a); Levin et al. (2012). Esse ponto trata justamente da separação dos planos de controle (elemento controlador que decide como manipular o tráfego) e de dados (elementos que manipulam o tráfego de acordo com as decisões estabelecidas pelo plano de controle). Essa dissociação entre o *hardware* de encaminhamento e as decisões de controle pode ser considerada a ideia básica do paradigma SDN, pois ela influencia diretamente as demais características. Utilizando essa abordagem, o controlador toma para si a responsabilidade de toda e qualquer decisão sobre como manipular os dados, deixando a cargo de *switches* e roteadores apenas a tarefa de encaminhamento desses dados de acordo com a decisão estabelecida. Segundo Zilberman et al. (2015), a separação dos planos não é um conceito novo, mas foi popularizado com o advento do paradigma.

Uma segunda característica chave do paradigma SDN é a questão da centralização do controle e da visão da rede. Segundo Macedo et al. (2015), SDNs são caracterizadas pela existência de um sistema de controle programável - em *software* – e centralizado que é capaz de controlar os mecanismos do plano de dados através de uma interface de programação bem definida. Em redes SDN, a inteligência da rede é logicamente centralizada no controlador SDN, que mantém uma visão global da rede, simplificando a aplicação de políticas, a configuração e a evolução da rede (LARA; KOLASANI; RAMAMURTHY, 2014); (KREUTZ et al., 2015).

Outro ponto chave trata da questão de programabilidade da rede, segundo Nunes et al. (2014a), a separação dos planos de controle e dados simplifica drasticamente o gerenciamento da rede e permite a inovação e a evolução da mesma. Em contraste, as redes tradicionais são formadas por um grande conjunto de protocolos, que evoluem lentamente e são difíceis de gerenciar, devido à configuração de baixo nível dos componentes individuais, o que torna complicado qualquer tipo de modificação na rede (FEAMSTER; REXFORD; ZEGURA, 2014a). Através de uma programação de mais alto nível proporcionada pelo paradigma SDN, novos serviços podem ser implantados mais rapidamente, diminuindo os custos operacionais das redes.

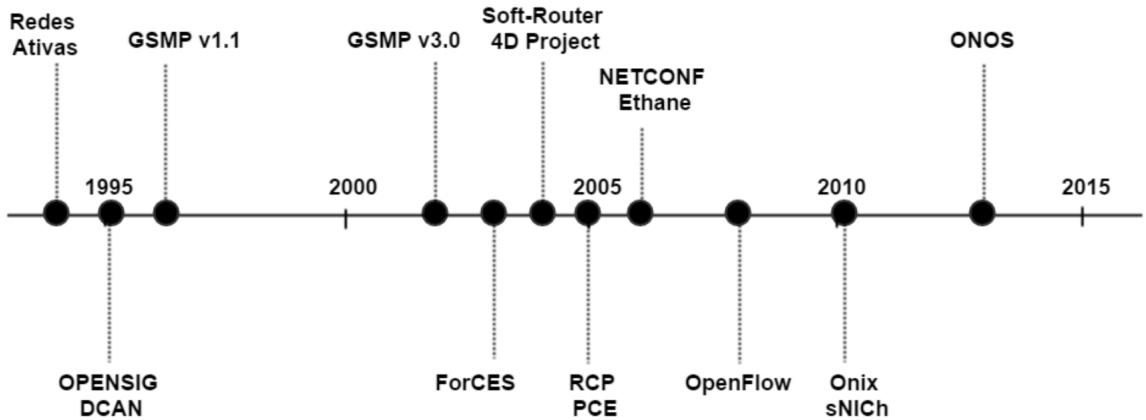
Uma última característica fundamental de SDN é a utilização de interfaces abertas para a programação dos dispositivos da rede. Com a separação dos planos de dados e de controle, os comutadores se tornam simples equipamentos de encaminhamento de pacotes, que podem ser programados através de uma interface aberta de programação bem definida (MACEDO et al., 2015); (NUNES et al., 2014b). Segundo Nunes et al. (2014b), quando implementado em padrão aberto, o paradigma SDN simplifica o projeto e a operação da rede, uma vez que as instruções são fornecidas unicamente pelo controlador, e não por múltiplos protocolos e dispositivos proprietários. Um exemplo disso é o padrão OpenFlow, discutido mais adiante neste trabalho.

Em resumo, considerando todos os aspectos comentados anteriormente, uma possível definição para SDN poderia ser a seguinte: SDN é um paradigma de redes de computadores, cuja implementação consiste na inserção de um elemento centralizado na rede, denominado controlador. O controlador é o elemento responsável por decidir como os dados (pacotes) serão manipulados pelos elementos comutadores, que por sua vez, irão apenas se concentrar na tarefa de manipular os dados de acordo com as decisões tomadas pelo controlador. Dessa forma, passa a existir uma clara separação entre o plano de controle e o plano de dados, o que simplifica o gerenciamento e a configuração da rede e, principalmente, permite a flexibilização da rede, a evolução da rede e o desenvolvimento de novos serviços, tudo isso através de uma programação de alto nível dos dispositivos comutadores por meio de uma interface aberta bem definida entre esses dispositivos e o elemento de controle programável.

2.2 Breve histórico

A Figura 1 retrata de forma resumida uma linha do tempo da evolução sobre as redes definidas por *softwares*, por meio do desenvolvimento de projetos anteriores e tecnologias e que fundamentaram como base para o atual desenvolvimento do paradigma SDN. Ao longo dessa seção, são discutidas essas tecnologias.

Figura 1 – Linha cronológica do desenvolvimento do paradigma SDN (NUNES et al., 2014b).



O termo original (*Software-Defined Networks* - SDN) foi utilizado pela primeira vez no ano de 2009, o artigo de Feamster, Rexford e Zegura (2014b) que trata sobre OpenFlow da Universidade de *Stanford*, que fora lançado dois anos antes. Apesar de a nomenclatura SDN ser relativamente novo e o conceito por trás dela ter evoluído, o pilar fundamental para a concepção do princípio denota a década de 1990.

Até então, as redes tradicionais operavam com os planos de controle e de dados combinados dentro de cada nó da rede. Essas redes transportavam dados passivamente, de um sistema final para outro, e essa transferência era a mais simples possível, sem qualquer tipo de modificação da unidade de informação. O papel da computação dentro das redes era limitado a tarefas pontuais como, por exemplo, o processamento dos cabeçalhos dos pacotes, que antecedia ao encaminhamento dos mesmos (TENNENHOUSE et al., 1997). Além disso, nessa abordagem, apenas a política de encaminhamento era definida e o único jeito de fazer ajustes nessa política era através de mudanças na configuração dos dispositivos (SEZER et al., 2013). Tais fatos contribuem para uma restrição da capacidade de configuração e da evolução no tocante a extensão das redes tradicionais, tornando-se um fator limitador importante ao longo de décadas. De forma direta, as redes passaram a ter uma evolução lenta, com pouco espaço para novas inovação tecnológicas, propiciando uma certa estagnação tanto na criação de novos protocolos e serviços quanto em sua arquitetura.

Para tentar diminuir esses problemas, várias tentativas foram aplicadas e desenvolvidas ao longo dos anos, tais como a adição de mais recursos de programações a redes de computadores. O objetivo principal dessas tentativas era tornar as redes mais dinâmicas, permitindo assim a experimentação e inovação de novos desenvolvimentos de protocolos e serviços.

Nesse viés nasceu uma das primeiras iniciativas, denominada de Redes Ativas (*Active Networks* - AN). Concebida entre os anos de 1994 e 1996, e posteriormente publicada no trabalho de pesquisa Tennenhouse et al. (1997), o termo Redes Ativas representa uma forma introdutória a programação da rede. Em sua pesquisa, Macedo e outros (2015) afirmam que a criação e utilização de virtualização de computadores, como *hypervisors* e máquinas virtuais, influenciaram diretamente em seu desenvolvimento. A ideia central dessa proposta consiste no fato de que *switches* ou roteadores em uma Rede Ativa passam a ter a capacidade de realizar cálculos não mais específicos sobre os pacotes, ou ainda, alterar o conteúdo dos mesmos (TENNENHOUSE et al., 1997; KREUTZ et al., 2015). Para realizar essas tarefas, as Redes Ativas sugerem duas abordagens distintas: cápsulas e *switches* programáveis. Na abordagem de cápsulas, os pacotes são substituídos por pequenos programas que são encapsulados em quadros de transmissão e executados em cada nó ao longo do caminho percorrido (TENNENHOUSE et al., 1997). Na abordagem de *switches* programáveis, por outro lado, o formato dos pacotes (ou células) não é modificado. Além disso, os dispositivos comutadores devem ser capazes de suportar o *download* de programas com instruções específicas sobre como processar os pacotes (KREUTZ et al., 2015; TENNENHOUSE et al., 1997). As redes Ativas trouxeram contribuição

a ideia de SDN que se tem hoje. Elas foram pioneiras na utilização da programação de redes, permitiram a inovação e a criação de novos serviços em ambientes produtivos. Também foram as primeiras que trataram da questão de virtualização da rede, além de realizarem a demultiplexação personalizada de programas em *software* de acordo com campos específicos do cabeçalho dos pacotes (FEAMSTER; REXFORD; ZEGURA, 2014a). Infelizmente, a iniciativa de Redes Ativas nunca conseguiu reunir uma grande massa crítica e acabou não sendo implantada em redes de produção de larga escala devido, principalmente, a questões de segurança e desempenho dos roteadores, que não eram capazes de processar a mesma quantidade de pacotes quando comparados a roteadores tradicionais (NUNES et al., 2014a; MACEDO et al., 2015).

Outra iniciativa proposta foi a do grupo de trabalho OpenSignaling (OPENSIG), que iniciou, em 1995, uma série de discussões e *workshops* com o objetivo de tornar redes como ATM (*Asynchronus Transfer Mode*), *Internet* e redes móveis mais abertas, extensíveis e programáveis. Esse grupo defendia a ideia de que a separação entre o *hardware* de comunicação e o *software* de controle era necessária, mas difícil de realizar, devido, principalmente, a *switches* e roteadores verticalmente integrados, cuja natureza fechada tornou praticamente impossível o rápido desenvolvimento de novos serviços e ambientes de rede (NUNES et al., 2014b). A proposta dessa iniciativa estava centrada na implantação do acesso ao *hardware* de rede através de interfaces de rede abertas e programáveis, o que coincide com uma das características do atual padrão SDN. Sendo assim, novos serviços poderiam ser criados através de programação em ambientes distribuídos (NUNES et al., 2014b). Nessa proposta também estavam elencados quatro níveis de abstração: elemento de rede virtual, elemento físico, serviços de valor adicionado e serviços genéricos. Foi determinado pelo grupo de trabalho serviços e recursos para cada um desses níveis, criando posteriormente interfaces para o controle desses níveis (NUNES et al., 2014b). Como resultado final, o grupo elaborou a especificação de um novo protocolo, o GSMP (*General Switch Management Protocol*), de uso geral para controle de um *switch*. Esse protocolo, de forma resumida, permitia que um controlador liberasse e estabelecesse conexões através do *switch*, gerenciando portas, informações de configuração, estatísticas e reserva de recursos do *switch* (DORIA, 2002; NUNES et al., 2014b).

Com a criação do protocolo GSMP, houve uma primeira tentativa de se criar uma entidade base de controle de equipamentos de comutação, a qual estivesse separada fisicamente. Em meados dos anos 1990 uma outra tentativa de separação dos planos fora desenvolvida pela ATM. Foi denominado de DCAN (*Devolved Control of ATM Networks*) foi um projeto que tinha o objetivo principal, desenvolver a infraestrutura para gerenciamento e controle escalável das redes ATM. Sua premissa de separação dos planos vai ao encontro da premissa do OPENSIG, inclusive na criação de um protocolo minimalista entre o gerenciador e a rede (NUNES et al., 2014b).

Contudo, até por volta dos anos 2000, os esforços se centravam na melhoria da programabilidade das redes iniciada pela proposta de Redes Ativas. Somente a partir de então, uma série de esforços que visavam a separação dos planos de dados e de controle começou a emergir.

Foi o caso da arquitetura ForCES (*Forwarding and Control Element Separation*). Inicializada em 2003, ela definiu dois elementos funcionais, o elemento de controle (CE) e o elemento de encaminhamento (FE), o que de fato passou a separar os planos. ForCES é uma arquitetura de inúmeros recursos, capaz de atender a topologias compostas por grandes quantidades de CEs e FEs, além de possuir uma interface aberta e programável entre os dois planos e um arcabouço de módulos que implementam tarefas de controle específicas (MACEDO et al., 2015).

Nos anos seguintes, foram desenvolvidas outras arquiteturas como a *Routing Control Platform* (RCP) e a *Soft-Router*, assim como o protocolo *Path Computation Element* (PCE), que trouxeram como contribuição principal o controle lógico centralizado da rede, através de uma interface aberta para o plano de dados (FEAMSTER; REXFORD; ZEGURA, 2014a). A arquitetura *Soft-Router* usa a API de programação da ForCES que possibilita que controlador instale entradas na tabela de fluxos do plano de dados, removendo a funcionalidade de controle dos roteadores (FEAMSTER; REXFORD; ZEGURA, 2014a). Já a RCP utiliza o protocolo padrão *Border Gateway Protocol* (BGP) para instalar entradas na tabela de fluxo nos roteadores tradicionais (CAESAR et al., 2005).

Ainda em meados do ano 2000, surgiu o projeto Ethane proposto por Casado et al. (2009) e desenvolvido pelo departamento de Ciência da Computação da Universidade de *Stanford*. O Ethane apresentou um projeto de arquitetura de redes corporativas que se baseava em uma solução queqe envolvia um controlador logicamente centralizado, que atuaria como gerenciador de políticas de encaminhamento, de acesso à rede e de segurança. Resumidamente, redes Ethane são compostas por duas entidades: um *switch* Ethane, que consiste em uma tabela de fluxo e um canal seguro para comunicação com o controlador e um controlador centralizado que decide se um pacote deve ser encaminhado (NUNES et al., 2014b). O Ethane reduziu os *switches* a simples tabelas de fluxo que são manipuladas pelo controlador com base em políticas de segurança de alto nível (FEAMSTER; REXFORD; ZEGURA, 2014a). Nesta arquitetura, o controlador tem o conhecimento da topologia global da rede e realiza a computação das rotas apenas para fluxos autorizados, através da escrita das ações devidas na tabela de fluxo dos *switches*. Ou seja, quando um pacote é recepcionado e não há registro de um fluxo que correspondente a esse pacote, ele é encaminhado ao controlador, que, por sua vez, decidirá o que fazer (CASADO et al., 2009).

Não obstante, as redes Ethane adotaram um conceito que passou a ser amplamente utilizado a partir de então: o conceito de fluxo. Segundo ele, um fluxo é um conjunto de pacotes que possuem campos idênticos de cabeçalho. Ou seja, quando pacotes diferentes chegam ao *switch*, mas possuem campos de cabeçalho idênticos, eles serão tratados da mesma forma, uma vez que serão considerados integrantes de um mesmo fluxo. Desta forma, apenas uma única política de decisão pode ser criada para cada fluxo, e não mais para cada pacote individual que chega à rede (CASADO et al., 2009).

Para alguns autores, a história do paradigma SDN se inicia aqui, com a criação do projeto Ethane, uma vez que grande parte das ideias elaboradas nesse projeto e vários dos conceitos

adotados serviram como base para o desenvolvimento do paradigma SDN. Além disso, o projeto serviu como base para a elaboração do projeto que tornou possível, de fato, a implementação do paradigma SDN: o protocolo OpenFlow.

O protocolo OpenFlow foi desenvolvido também na Universidade de *Stanford*, assim como seu antecessor, e publicado em 2008, no trabalho de McKeown e outros (2008). Na prática, foi adotado as premissas de SDN previamente discutidas e seguiu a estrutura adotada pelo projeto Ethane. A sua grande contribuição foi ter conseguido trazer toda a ideia de programabilidade de redes e de separação entre controle e dados para um desenvolvimento real e prático do paradigma SDN. Por ser uma solução menos complexa, que não requer grandes modificações nos dispositivos comutadores, foi possível tornar atrativa não somente para a comunidade acadêmica, mas também para a indústria (KREUTZ et al., 2015). A arquitetura e outras características do protocolo OpenFlow estão detalhadas em uma seção mais adiante.

Concluindo, o paradigma SDN foi evoluindo, gradativamente, fundamentado por projetos e ideias já propostas, principalmente as ideias que tratavam de aspectos como separação dos planos e controle centralizado. Foi essa evolução que transformou a SDN na iniciativa de sucesso que ela é hoje.

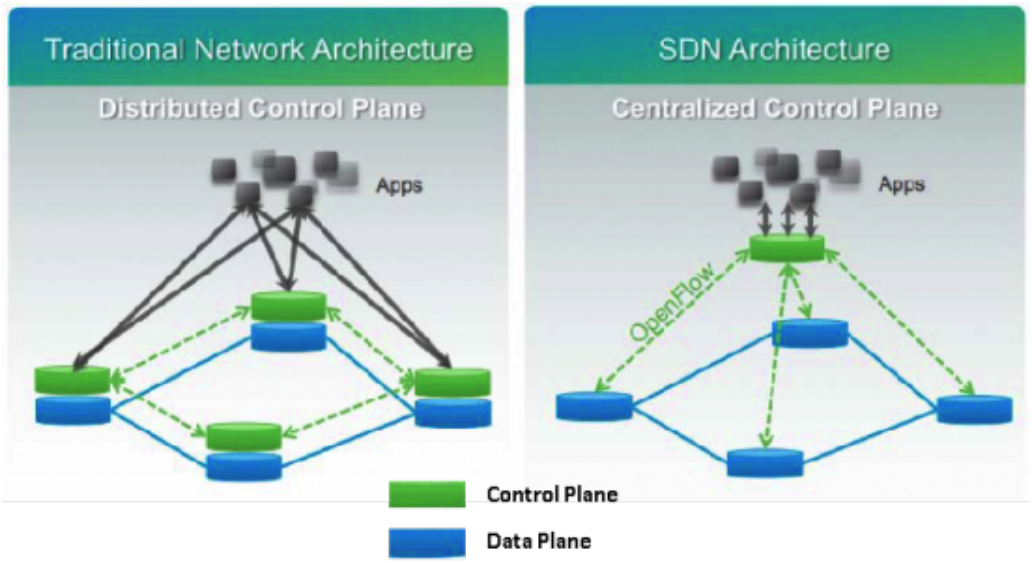
2.3 Componentes de uma arquitetura

Considerando o que já foi discutido nas seções anteriores, poderia-se dizer, de forma simplificada, que uma Rede Definida por *Software* é caracterizada pela existência de duas entidades principais: um elemento controlador, que tem o poder de determinar o modo como os dados devem ser manipulados em uma rede; e os elementos de encaminhamento, que são os equipamentos de comutação que devem manipular os dados de acordo com o que foi definido pelo elemento controlador. Contudo, uma SDN não pode ser caracterizada apenas dessa maneira. Existem outros elementos, físicos e abstratos, que desempenham importantes papéis em uma estrutura típica do paradigma. A Figura 2 ilustra uma arquitetura SDN típica, composta por seus principais elementos. Nas subseções a seguir, esses elementos são apresentados com detalhes.

2.3.1 Dispositivos endpoints

São os equipamentos localizados na ponta das redes, tais como computadores, servidores, impressoras, *notebooks* e demais dispositivos que acessam e utilizam os serviços providos pela rede. Eles são elementos não programáveis, contudo, indiretamente sofrem influência das decisões tomadas pelo controlador.

Figura 2 – Arquitetura SDN típica. (SEZER et al., 2013)



2.3.2 Equipamentos de comutação programáveis

São os equipamentos de comutação similares aos das redes tradicionais, como *switches* e roteadores – virtuais ou físicos – que, ao serem inseridos em uma arquitetura SDN, se tornam simples dispositivos de encaminhamento de pacotes, uma vez que toda e qualquer tarefa de decisão ou controle foi retirada de seu escopo e atribuída ao controlador, devido à separação física dos planos de dados e de controle. Portanto, resta a esses dispositivos apenas a tarefa do plano de dados, que é a de encaminhar pacotes.

2.3.3 Controlador

O controlador é o dispositivo da rede que monitora e modifica o comportamento dos elementos programáveis da rede, como *switches* e roteadores, além dos elementos finais de acesso à rede. É também o componente principal do plano de controle, e pode estar associado a outros controladores. Em uma SDN, o controlador utiliza um conjunto de APIs para interagir com os elementos programáveis, que são geralmente os dispositivos comutadores. O controlador envia seus comandos de controle para esses elementos programáveis através do canal de controle seguro, utilizando uma API de baixo nível, que geralmente é dependente de hardware e tende a ter um grau limitado de abstração (MACEDO et al., 2015). Para facilitar a programação da rede, o controlador age como se fosse um sistema operacional para ela, exportando uma interface de programação de alto nível para operadores e desenvolvedores que abstraia os detalhes de operação de cada componente e, além disso, que permita a execução simultânea de diferentes programas de controle. Além de ser denominado como sistema operacional de redes, o controlador também é tratado por alguns autores como *hypervisor* da rede, por muitas vezes permitir o particionamento da rede em redes virtuais.

Além da característica principal, que é estabelecer a comunicação com todos os elementos

programáveis de uma SDN, o controlador também fornece uma visão unificada do *status* da rede. Essa visão unificada (e centralizada) da rede, um dos pontos fortes do paradigma SDN, permite o desenvolvimento de análises detalhadas para determinar a melhor decisão operacional sobre como o sistema, como um todo, deve atuar. Essa visão global da rede acaba simplificando a tarefa de gerenciamento, a descoberta e resolução de problemas e a tomada de decisão. Além disso, é importante salientar que essa visão única e centralizada da rede fornecida pelo controlador é uma visão lógica, não exigindo, necessariamente que o controlador seja um componente concentrador da rede, fisicamente localizado em um ponto específico e único do sistema. Essa visão pode ser abstraída através da implementação de soluções de forma distribuída, quer seja pela utilização de múltiplos controladores resolvendo problemas específicos em pontos distintos da rede ou pela divisão dos elementos da visão entre domínios diferentes em um mesmo ponto da rede. Ademais, um único controlador centralizado representa um único ponto de falha em toda a rede, contudo protocolos como o OpenFlow permitem a conexão de múltiplos controladores a um *switch*, o que garantiria a redundância no plano de controle, no caso em que um evento de falha no controlador principal venha a acontecer.

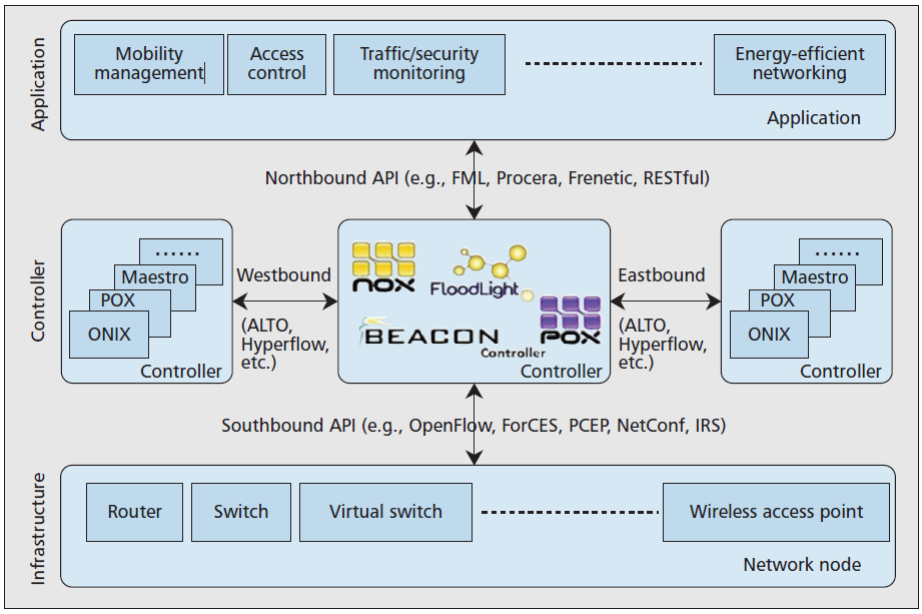
Em geral, o controlador é responsável pela execução de códigos de programas de controle que podem ser desenvolvidos utilizando-se linguagens de alto nível, que posteriormente serão traduzidos pelo próprio controlador em ações que podem ser enviadas a cada elemento da rede. Basicamente, aqui reside a diferença funcional entre o código do programa, que determina o que será feito, e o controlador, que transforma o código de alto nível de uma linguagem de programação em comandos entendidos pela API, que serão enviados aos comutadores. Daí o conceito de sistema operacional da rede atribuído ao controlador.

Geralmente, o controlador é composto também por duas interfaces, *Southbound* e *Northbound*, cada qual responsável por estabelecer a relação do controlador com algum elemento da arquitetura. As subseções seguintes abordam esses dois elementos. A relação entre o controlador e essas interfaces está ilustrada na Figura 3.

2.3.4 API Northbound

As interfaces *Northbound*, estão localizadas no nível mais acima da plataforma de controle, são responsáveis por controlar a interação entre as aplicações de controle e o controlador. Sua função principal é traduzir os requisitos das aplicações de gerenciamento em instruções de baixo nível para os dispositivos da rede e transmitir dados e estatísticas sobre a rede, que foram geradas nos dispositivos da rede e processadas pelo controlador (MCKEOWN et al., 2008). A API *Northbound* pode ser vista como uma interface de programação que permite que os programas desenvolvidos em linguagens de alto nível possam ser capazes de controlar a rede (Figura 3). Comparadas às APIs *Southbound*, as interfaces *Northbound* possuem um nível maior de abstração. A expressão "norte"refere-se à direção controlador/aplicação, em cuja relação há a comunicação entre os serviços e aplicações que devem ser executados (no

Figura 3 – Relação entre o controlador e as interface *Northbound* e *Southbound* (SEZER et al., 2013).



nível mais alto da arquitetura), e o controlador, que deve ser capaz de "compreender"o código para posteriormente transformá-lo em comandos válidos para a rede. Diferentemente da API *Southbound*, que já possui um padrão amplamente aceito OpenFlow, não existe uma definição de uma API *Northbound* padrão para SDN. A maioria dos controladores (Floodlight, NOX, Ryu, etc...) define suas próprias API *Northbound* (MACEDO et al., 2015).

2.3.5 API Southbound

As interfaces *Southbound*, localizadas no nível mais inferior da plataforma de controle, são as responsáveis por controlar a interação entre os dispositivos de encaminhamento e o controlador. A expressão "sul"refere-se à direção controlador/*switch*, cuja relação há o estabelecimento de comunicação do controlador para um nível mais inferior da arquitetura, que são os equipamentos físicos da rede. O protocolo OpenFlow pode ser visto como exemplo de uma API *Southbound*, uma vez que ele implementa as interações e a comunicação entre os dispositivos de comutação e o controlador da rede (Figura 3). A grande maioria dos controladores suporta apenas o OpenFlow como API *Southbound*, mas alguns como o Onix e o *FloodLight* suportam mais APIs ou *plugins* de protocolo, contudo há outras APIs, como OVSDB, ForCES e OpFlex, que adicionam novas funcionalidades. Por exemplo, a interface OVSDB permite aos elementos de controle a configuração túneis e gerenciamento de filas, criação de diversas instâncias de comutadores virtuais e adicionar políticas de QoS nas interfaces. Por isso, ela é uma interface complementar ao OpenFlow para o uso com *Open vSwitch* (MACEDO et al., 2015).

2.3.6 Aplicação

Como os controladores são considerados de fato os sistemas operacionais em redes SDN, os códigos em *softwares* são criados para definir novas funcionalidades e serviços, e que serão executadas no controlador, podem ser considerados como sendo as aplicações que são executadas sobre a rede física. Em outras palavras, as aplicações de rede implementam o controle lógico que será traduzido em comandos de configuração das regras no plano de dados, determinando o comportamento dos dispositivos comutadores. Existem diversas aplicações de redes tradicionais que podem ser implementadas com SDN, tais como roteamento ou *firewall* e balanceamento de carga. Todavia, novos serviços podem ser implementados agora em mais alto nível de abstração, como economia de energia e monitoramento das redes. Em uma seção adiante, serão abordadas algumas das aplicações possíveis em SDN.

2.3.7 Virtualizador de rede

Com a criação de SDN, se tornou possível identificar padrões diferentes de pacotes que passam pelo comutador, que são posteriormente classificados e segmentados em fluxos. Desta forma, tornou-se possível também o tratamento diferenciado para diferentes fluxos. A exemplo da separação feita em uma rede acadêmica entre os fluxos tradicionais de pesquisa e os de produção, que pode ser realizada agora, com SDN, sem trazer nenhum tipo de prejuízo para a rede ou para o serviço relacionado a ambos os fluxos (MCKEOWN et al., 2008). Essa divisão "virtual"no tratamento de diferentes padrões de fluxo é realizada pelo virtualizador de rede. Sua responsabilidade é gerenciar as redes virtuais, que além de prover a separação das visões é capaz também de prover uma divisão dos recursos físicos entre essas visões. Cada rede virtual formada pelo divisor pertence a uma visão diferente e possui seu próprio *software* de plano de controle. Sendo assim, é possível coexistir diversos controladores funcionando sobre uma mesma rede física, entretanto virtualmente separados. Os recursos de rede podem ser virtualizados e apresentados de forma isolada para cada visão, que por sua vez só terá acesso aos elementos e recursos da rede reservados para a sua fatia da rede física.

2.4 O protocolo OpenFlow

A premissa mais importante por trás do paradigma de Redes Definidas por *Software* é a capacidade de controlar, à distância, a tarefa de encaminhamento de pacotes, através de uma interface de programação bem definida (MCKEOWN et al., 2008). Foram desenvolvidos, ao longo dos anos, alguns protocolos capazes de implementar essa comunicação entre dispositivos inseridos em uma SDN. Contudo, um desses protocolos está associado ao paradigma desde a sua concepção, e, de fato, foi o que tornou possível a implementação prática da SDN que é conhecida atualmente: o protocolo OpenFlow (LARA; KOLASANI; RAMAMURTHY, 2014).

O protocolo OpenFlow, desenvolvido na Universidade de *Stanford*, e proposto no trabalho de McKeown et al. (2008) em 2008, é um padrão aberto para Redes Definidas por *Software* que tem como principal objetivo permitir que aplicações em *software* possam programar de forma mais simples as tabelas de fluxos dos dispositivos de comutação de pacotes presentes nas redes, através de interfaces abertas de programação. Tal programação influencia no controle desses dispositivos por uma entidade centralizada, o controlador, o que ocasiona, na prática, a separação dos planos de dados e de controle. Nesse quesito, é importante ressaltar a diferença entre o paradigma SDN e o protocolo OpenFlow, que muitas vezes são considerados sinônimos. Enquanto SDN é um paradigma que consiste na concepção de separação dos planos, o OpenFlow descreve o modo como o *software* controlador e os *switches* devem se comunicar em uma arquitetura SDN (LARA; KOLASANI; RAMAMURTHY, 2014).

Segundo McKeown et al. (2008), o OpenFlow padronizou a forma de comunicação entre os dispositivos de encaminhamento e o controlador. Dessa forma, o controlador, que é composto por um código em *software* de alto nível, desenvolvido em uma linguagem de programação determinada, pode enviar uma série de operações em formato de comandos OpenFlow ao comutador, que serão entendidas e processadas graças a essa padronização na forma de comunicação. Tais comandos OpenFlow são capazes de programar a tabela de fluxos dos *switches* quando o protocolo está habilitado.

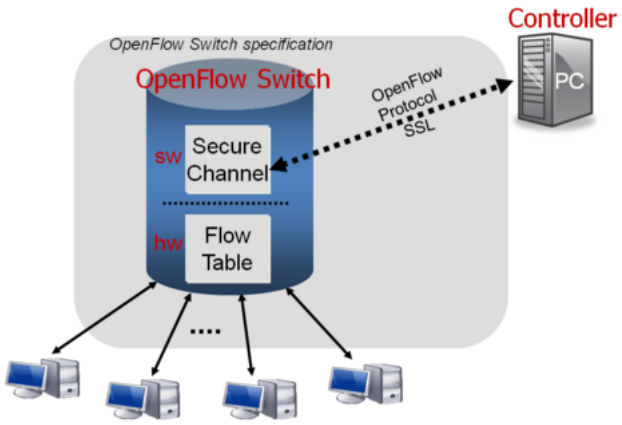
Para todos os casos, neste trabalho, a versão de referência do protocolo OpenFlow utilizada nos exemplos práticos é a versão 1.3. Esta versão foi escolhida porque é a versão do protocolo mais utilizada e suportada (LARA; KOLASANI; RAMAMURTHY, 2014).

2.4.1 Arquitetura do protocolo OpenFlow

Quando se habilita o protocolo OpenFlow em um *switch*, a arquitetura do dispositivo passa a ser constituída por três elementos básicos, como demostrado e ilustrado na Figura 4: (i) uma tabela de fluxos, na qual existe, para cada entrada da tabela, uma ação associada que define como o *switch* vai processar esse fluxo relacionado; (ii) um canal seguro, que liga o *switch* ao controlador, e por onde passam os comandos e os pacotes trocados; e (iii) o protocolo OpenFlow, que fornece uma interface padrão de comunicação entre controlador e *switch*. Devido à especificação dessa interface padrão, as entradas da tabela de fluxos podem ser programadas externamente pelo controlador, que envia comandos através do canal seguro (MCKEOWN et al., 2008).

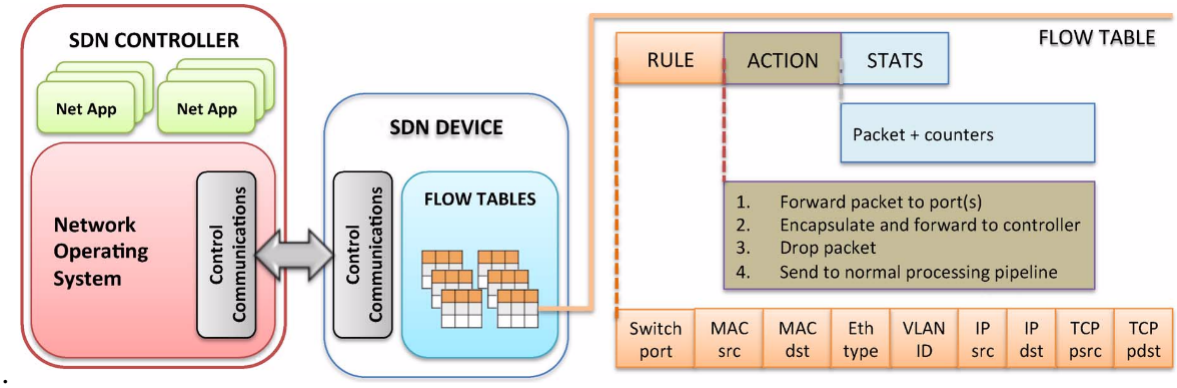
A tabela de fluxos é composta por diversas entradas, e cada uma dessas entradas da tabela possui três atributos: (i) um campo composto por vários campos de cabeçalho, que especificam o fluxo associado a essa entrada; (ii) um campo de ações, que determinam como os pacotes desse fluxo serão processados; e (iii) um campo com contadores, que mantém os registros de números de pacotes e bytes de cada fluxo, bem como tempo decorrido desde a chegada do último pacote associado ao fluxo (Figura 5). Em resumo, a associação entre um determinado fluxo e um

Figura 4 – Um *switch* OpenFlow se comunica com um controlador através de um canal seguro, usando o protocolo OpenFlow (MCKEOWN et al., 2008).



conjunto de ações associado ao fluxo formam uma entrada na tabela de fluxos (MCKEOWN et al., 2008). Não obstante, cada entrada na tabela de fluxos de um *switch* OpenFlow pode ser armazenada em memória local. Tipicamente, é utilizada memória memória TCAM (*Ternary Content-Addressable Memory*) ou SRAM (*Static Random Access Memory*), ou uma memória na qual os bits podem ser representados por valores iguais a "zero", "um"ou "não importa", esse último indicando que ambos os valores anteriores são aceitáveis naquela posição (LARA; KOLASANI; RAMAMURTHY, 2014; XIA et al., 2015; MACEDO et al., 2015).

Figura 5 – Entrada da tabela de fluxos: atributos que determinam o fluxo (ou regra) associado, ações e contadores (KREUTZ et al., 2015)



Os campos de cabeçalho de uma entrada da tabela, que são semelhantes aos campos de cabeçalho típicos de um pacote IP, são associados a valores previamente definidos ou padrão. Esses valores serão comparados aos valores dos campos de cabeçalho de cada pacote que chega ao *switch*, com o objetivo de determinar se o pacote pertence ao fluxo definido por essa entrada da tabela. Se os valores dos campos do pacote forem iguais aos valores dos campos definidos na entrada da tabela, há um *match*, o que significa que o pacote pertence àquele fluxo definido na entrada da tabela. Se algum dos valores dos campos do pacote possuir valores diferentes com o valor do respectivo campo da entrada da tabela, não haverá um *match* para esse fluxo. Desta forma, o pacote continuará a ser comparado com as demais entradas da tabela, até sofrer

um *match* ou, no caso de insucesso em todas as comparações, será enviado ao controlador, que decidirá o que fazer com ele. A parte inferior (em laranja) da Figura 5 mostra quais são os atributos utilizados nas entradas da tabela de fluxo para a realização de *matches*, implementados pela versão 1.0.0 do protocolo OpenFlow.

Já as ações determinam como o *switch* deve proceder com os pacotes correspondentes a um determinado fluxo. Cada fluxo pode ser associado a nenhuma, uma ou várias ações. Entretanto, diversas implementações OpenFlow existentes são capazes de realizar uma única ação. As possíveis ações que podem ser tomadas por um *switch* OpenFlow em relação aos pacotes são: (i) o encaminhamento, que consiste no repasse do pacote para uma porta específica do *switch*, para todas as portas, de volta para a porta de entrada, ou para o controlador; (ii) o enfileiramento, que consiste em enviar um pacote para uma fila associada a uma porta específica, geralmente para prover um serviço básico de QoS; (iii) o descarte, que consiste em descartar pacotes explicitamente ou, caso não exista ação especificada para um determinado fluxo, descartar todos os pacotes que casarem com esse fluxo; e (iv) a modificação de campo de cabeçalho, que consiste na mudança do valor de algum campo do cabeçalho do pacote que chegou. Contudo, para versões mais recentes, novas ações e operações foram adicionadas. Essas modificações são apresentadas na subseção a seguir.

2.4.2 Versões do protocolo OpenFlow

Atualmente, estão disponíveis diferentes versões do protocolo OpenFlow, sendo que a mais recente delas, a versão 1.5, data do final de 2014 (MACEDO et al., 2015). Cada uma delas, gradativamente, vem acumulando modificações, melhorias ou até novas funcionalidades ao protocolo. Entretanto, a versão mais amplamente utilizada e suportada, no momento, é a versão 1.3 (LARA; KOLASANI; RAMAMURTHY, 2014).

As primeiras versões do protocolo OpenFlow (0.2, 0.8 e 0.9) datam de 2008 e 2009. Essas versões eram instáveis, que, inclusive, já foram descontinuadas. Somente a partir de dezembro de 2009, foi lançada a primeira versão estável do protocolo OpenFlow, que é a versão 1.0. Nesta versão, o controlador dispõe de apenas uma única tabela de fluxos em cada *switch*, e as operações de *match* podem ser realizadas sobre os 12 campos de cabeçalho apresentados anteriormente na parte inferior da Figura 5: porta física de entrada, endereços MAC de origem e destino, *ethertype*, ID da VLAN, prioridade da VLAN, endereços IP de origem e destino, protocolo IP, IP *Type of Service* e portas TCP/UDP de origem e destino. O número de ações também era resumido a apenas quatro: encaminhamento, enfileiramento, descarte e modificação de campos de cabeçalho.

A versão 1.1, apresentada em fevereiro de 2011, passou a suportar operações em múltiplas tabelas de fluxos. Outras grandes novidades dessa versão foram o suporte a *match* em campos MPLS (*Multiprotocol Label Switching*) e a tabela de grupos, que possibilitou operações em comum de múltiplos fluxos fossem agrupadas e realizadas. Foi aumentado o número de ações (alteração de TTL, agrupamento, cópia de campos, operações de QoS e *push/pop* de *tags* de

cabeçalho), o que aumentou o poder e a flexibilidade do protocolo no reconhecimento de fluxos e na realização de operações (LARA; KOLASANI; RAMAMURTHY, 2014).

Em dezembro de 2011 foi lançada a versão 1.2 do protocolo OpenFlow. A principal funcionalidade trazida por essa versão foi a possibilidade de conectar um *switch* a múltiplos controladores simultaneamente, o que tornou possível tarefas como recuperação de falhas e balanceamento de carga entre controladores. Outras características adicionadas por essa versão foram o suporte aos protocolos IPv6 e ICMPv6 (MACEDO et al., 2015).

A versão 1.3 do protocolo OpenFlow, que foi lançada em junho de 2012, trouxe melhorias como o controle da taxa de pacotes através de *meters* por fluxo, a habilitação de conexões auxiliares entre o *switch* e o controlador, o suporte aos cabeçalhos de extensão do IPv6 e o suporte para comunicação encriptada por TLS. Esta foi uma versão amplamente aceita pela comunidade, tanto pela sua facilidade de uso quanto pela facilidade de prototipação de novas funcionalidades. Até por isso, a ONF havia escolhido essa versão para a validação de novos recursos (FERNANDES E.L., 2013).

Em outubro de 2013 foi lançada a versão 1.4 do protocolo OpenFlow, que trouxe uma grande quantidade de melhorias e novidades. As principais foram: o suporte a execução de ações conjuntas, denominadas *bundles*, por meio de uma única operação; o monitoramento de fluxos em tempo real por parte do controlador para detecção de alterações nos *switches*; a exclusão de regras de menor importância para a inserção de novas regras quando os *switches* operam com a tabela de fluxos totalmente ocupada; a comunicação ao controlador de que a tabela de fluxos está próxima do seu limite (*vacancy events*); a mudança da porta TCP padrão usada pelo protocolo (da 6633 para a 6653); dentre outras mudanças (NUNES et al., 2014b).

Por fim, a última versão do protocolo OpenFlow lançada até então, a 1.5, que data de dezembro de 2014, traz ainda mais melhorias ao protocolo. As principais são: o suporte a *egress table*, que habilita o processamento no contexto da porta de saída; a extensão das estatísticas das entradas de fluxo; o *match* por *flags* TCP como ACK, SYN e FIN; o monitoramento das conexões com os controladores; o agendamento de *bundles*; o processamento de outros tipos de pacotes, como IP e PPP, e não apenas de pacotes *Ethernet* e MPLS, como era até então; dentre outras melhorias (LARA; KOLASANI; RAMAMURTHY, 2014).

2.4.3 Suporte ao protocolo OpenFlow

Diversas empresas de tecnologia de rede vem, ao longo dos anos, implementando e habilitando o padrão OpenFlow em seus equipamentos, tais como: *switches*, roteadores e pontos de acesso. Algumas empresas como Cisco, Dell, Extreme, HP, Huawei, IBM, Juniper, LinkSys, dentre outras, já disponibilizaram modelos de seus equipamentos com suporte a OpenFlow (NUNES et al., 2014b; MACEDO et al., 2015; LARA; KOLASANI; RAMAMURTHY, 2014; KREUTZ et al., 2015). Além dessas implementações em *hardware*, também existem as imple-

mentações em software. Algumas delas, como o *Open vSwitch*, foram desenvolvidas para serem executadas em um computador ou processador embarcado; outras foram desenvolvidas para placas como a NetFPGA; e outras ainda desenvolvidas para pontos de acesso, como o sistema Pantou/OpenWRT (KREUTZ et al., 2015; NUNES et al., 2014b).

2.5 Aplicações em SDN

Aspectos do paradigma SDN como a programabilidade de elementos comutadores e a visão logicamente centralizada da rede contribuem de forma decisiva para o desenvolvimento de novos serviços, bem como para a evolução daqueles já existentes. Essa flexibilidade trazida por SDN para a estruturação das redes tornou mais simples e prática a criação de serviços típicos de redes. Esta seção, apresenta diferentes tipos de aplicações que podem ser aplicadas em SDN.

Balanceamento de carga. O balanceamento de carga é um serviço muito comum em redes atualmente, uma vez que seu objetivo principal é alcançar o melhor uso possível dos recursos disponíveis, através da divisão das requisições em servidores replicados. Essa otimização dos recursos acarreta o aumento da vazão das tarefas, a diminuição do tempo de resposta e, consequentemente, a melhoria do desempenho. E ainda mais, traz vantagens em termos de escalabilidade e disponibilidade. Contudo, as soluções para balanceamento existentes hoje no mercado, apesar de serem extremamente eficientes, possuem uma certa limitação quanto à flexibilização do serviço de acordo com as características próprias do negócio. Além disso, balanceadores de carga proprietários são geralmente muito caros, o que inviabiliza sua implantação em muitas redes. Com SDN, o balanceamento de carga pode ser realizado diretamente pelos dispositivos de comutação da rede, sem a necessidade de se utilizar algum outro dispositivo específico para isso, bastando apenas o desenvolvimento de um código no controlador que execute tal serviço. Algumas soluções para esse serviço utilizam técnicas SDN de escrita de regras de encaminhamento ou de reescrita de campos de cabeçalho para realizar o balanceamento já no dispositivo comutador.

Gerenciamento de redes corporativas. Atualmente, grande parte das políticas de gerenciamento das redes é configurada individualmente em cada elemento da rede à medida que as redes vão crescendo e que são inseridos mais e mais dispositivos nessas redes, torna-se mais complicado estabelecer configurações de interesse na rede como um todo, uma vez que a complexidade para manter sincronizadas as configurações individuais dos equipamentos aumenta proporcionalmente ao crescimento da rede. Dessa forma, as políticas definidas raramente são modificadas. Com a adoção do paradigma SDN, a premissa de centralizar logicamente o plano de controle permite que uma visão global da rede seja gerada, o que simplifica as ações de configuração e a monitoração de fluxos de interesse (JARSCHEL et al., 2014). Mais ainda, com a variedade de informações obtidas através da rede, as políticas de gerenciamento podem ser adaptadas dinamicamente e de forma automática de acordo com o estado atual da rede,

além de serem repassadas para cada dispositivo, através de regras específicas que podem ser transcritas nas tabelas de fluxo de cada comutador, através de comandos únicos compreendidos pelos diferentes equipamentos.

Segurança de redes. Tradicionalmente, a segurança em redes se dá através da implantação de equipamentos físicos como *firewalls* e servidores *proxy*. Com a utilização de SDN, a segurança nessas redes pode ser aumentada, uma vez que o paradigma pode fornecer mais uma camada de segurança (antes de permitir que o tráfego adentre a rede), além de fornecer um conjunto de serviços em software capaz de melhorar a capacidade de defesa da rede. Mais detalhadamente, esses serviços podem prover novas formas de detecção e defesa de ataques em redes. Através da habilidade de coletar dados e estatísticas da rede, o paradigma SDN permite o monitoramento de todos os elementos da rede, a fim de detectar e analisar padrões de tráfego que possam estar associados a possíveis ameaças de segurança, em qualquer ponto da rede. Dessa forma, ataques simples ou de maior escala, como os de DDoS, podem ser detectados pela simples análise de padrões de tráfego, e podem ser tratados da forma mais adequada possível (XIA et al., 2015). Se ataques forem detectados, o próprio controlador pode instalar, dinamicamente, regras de encaminhamento que possam bloquear o ataque, através do descarte de pacotes. Se houver apenas suspeitas, o tráfego pode ser identificado e encaminhado para algum sistema de inspeção de pacotes. Outras técnicas de proteção como modificação aleatória de *hosts* internos, identificação de sobrecarga de tráfego, identificação de tráfego anômalo e segurança da própria infraestrutura também podem ser realizadas e programadas em uma SDN.

Redes sem fio. Segundo Kreutz et al. (2015), o atual plano de controle distribuído de redes sem fio é considerado de baixa qualidade no gerenciamento do espectro limitado, na alocação de recursos de rádio, no gerenciamento de interferências, na implementação de mecanismos *handover* e no desempenho do balanceamento da carga entre as células. Com SDN torna-se mais fácil o desenvolvimento e gerenciamento de diferentes tipos de redes sem fio. Já existem soluções SDN que visam prover camadas programáveis e flexíveis para essas redes. Algumas soluções, como a OpenRadio Bansal et al. (2012), propõem uma camada de abstração para a separação entre o protocolo *wireless* padrão e o hardware, permitindo que as camadas MAC sejam compartilhadas através de diferentes protocolos utilizando plataformas multi-core. Outras, como a Odin Suresh et al. (2012), utilizam o próprio protocolo padrão 802.11, mas realizam o isolamento lógico entre fatias da rede.

Gerenciamento de data centers. Atualmente, em TI, muitos serviços são altamente dependentes de data centers eficientes e escaláveis. E ao longo dos últimos anos, vem sendo registrado um aumento na demanda por esses serviços, o que torna crítico qualquer tipo de operação em data centers. Qualquer interrupção ou atraso nesses serviços pode acarretar prejuízos ou perdas severas em negócios (NUNES et al., 2014b). Com SDN, diversas questões clássicas em data centers podem ser resolvidas ou mais bem tratadas, tais como a migração ativa de redes, a otimização da utilização da rede, o monitoramento e detecção de problemas em tempo real, o uso

de mecanismos de segurança, evitar falhas iminentes, a adaptação de protocolos de transporte, a detecção de comportamentos anômalos na rede, o suporte a QoS, a economia de energia, dentre outras (KREUTZ et al., 2015). Além disso, SDN pode ser capaz de fornecer isolamento de tráfego entre diversos usuários, utilizando virtualização de comutadores para interligar máquinas de usuários independente de suas localizações físicas. Inclusive, foi utilizando SDN que o Google implementou o B4 (BOSSHART et al., 2014), um sistema que interconecta os data centers do Google ao redor do mundo, permitindo níveis de escalabilidade, tolerância a falhas e controle tais, que não poderiam ser alcançados por meio da arquitetura WAN existente.

Virtualização. A virtualização da rede é uma técnica difundida e já existente há algum tempo que permite que arquiteturas de rede heterogêneas compartilhem a mesma infraestrutura. Ela consiste em dividir a rede física em múltiplas instâncias virtuais que podem ser alocadas a diferentes usuários ou aplicações. Embora SDN e virtualização sejam entidades independentes e existam separadamente, a relação entre as duas abriu um novo campo para pesquisas na área. Um exemplo de benefício dessa relação é a criação de tecnologias por parte de SDN para a virtualização da rede, como, por exemplo, na criação de plataformas abstratas e equipamentos em software, como o *Open vSwitch*. Outro benefício trata do teste de aplicações SDN. A capacidade de separar os planos de dados e de controle torna possível o teste e o desenvolvimento de aplicações de controle SDN em ambientes virtuais, antes que as mesmas sejam implementadas em uma rede em operação. O desenvolvimento do ambiente Mininet, que virtualiza uma rede SDN com *switches*, *hosts* e controladores, dentro de uma máquina virtual, é um exemplo disso. Um último exemplo trata-se, de fato, da divisão de recursos em uma SDN. Enquanto a virtualização convencional utiliza túneis e campos VLAN e MPLS para realizar a configuração nada prática de um por um dos dispositivos da rede para a divisão dos recursos. O *FlowVisor* é um exemplo de software desenvolvido para realizar essa divisão de recursos e administrar a múltipla divisão entre as visões dentro de uma mesma rede (XIA et al., 2015).

Monitoramento e medição. A utilização de SDN provê à rede a capacidade de realizar certas operações de monitoramento e medição sem a utilização de equipamentos e protocolos adicionais e sem ocasionar *overheads* (JARSCHEL et al., 2014). Isso ocorre devido às características intrínsecas do paradigma, que são: coletar as informações sobre a rede, a fim de manter uma visão global de seu estado; e transferir essas informações para um controlador logicamente centralizado. Assim, a informação pode ser processada em software com o objetivo de fornecer dados que atuem como parâmetros de monitoramento. Dessa forma, o administrador da rede pode tomar decisões estratégicas dinamicamente, uma vez que, com os parâmetros de interesse, ele passa a ter uma visão do estado corrente da rede, além de reagir a algum eventual mau funcionamento que venha a ocorrer em equipamentos da rede.

Concluindo, existem várias outras aplicações, áreas e serviços que podem se beneficiar do paradigma SDN, tais como,VLANs, redes não IP, redes VoIP, *cloud computing*, dentre outras.

3

Trabalhos Relacionados

Foi realizada uma pesquisa nas bases *IEEE Xplorer Digital Library*[1], *Springer*[2], *Google Scholar*[3] e *ACM Digital Library*[4] a fim de encontrar trabalhos que apresentassem estudos sobre compartilhamento de banda em redes definidas por *software*. Foram utilizados os seguintes termos de busca: *"Bandwidth Allocation"*, *"Software Defined Network"*, *"Bandwidth Reservation"*, *"Elastic Resource Sharing"*, *"OpenFlow"*, *"Quality of Service"*. Para buscar trabalhos que fundamentassem o compartilhamento de banda em redes definidas por *software*, e assim ter um melhor entendimento sobre o assunto foram utilizados os seguintes termos de busca:*"Software Defined Network"* e *"Bandwidth Allocation"*. Para ter um melhor entendimento sobre o protocolo de comunicação *"OpenFlow"* foram utilizados os seguintes termos de busca:*"OpenFlow"*, *"Software Defined Network"* e *"SDN"*. As buscas foram realizadas com período definido entre 2007 e 2017, ou seja, nos últimos 10 anos. Após a análise dos títulos dos artigos não foi encontrado nenhum artigo que tratasse especificamente a mesma abordagem com os mesmos critérios sobre compartilhamento de banda em redes definidas por *softwares*. Apesar disso, foram detectados artigos que utilizam técnicas similares para outras finalidades. Além desses, foram encontrados outros artigos com potencial para auxiliar no entendimento sobre compartilhamento de banda em redes SDN.

3.1 Modelos de alocação de banda

O AlloCT-Sharing é um modelo de alocação de banda, proposto em (REALE; NETO; MARTINS, 2011), que utiliza uma estratégia de gerenciamento da largura de banda em redes multisserviços, fazendo sua alocação de modo oportunista. Foi proposta uma nova técnica que é aplicada em classes de tráfego prioritárias e não prioritárias. A ideia consiste em permitir que

[1] <http://ieeexplore.ieee.org/Xplore/home.jsp>
[2] <https://link.springer.com/>
[3] <https://scholar.google.com>
[4] <http://dl.acm.org/>

classes de tráfego de maior prioridade possam também utilizar a banda disponível e não utilizada para as classes de tráfego de menor prioridade. Como principal característica operacional, este modelo tem como objetivo permitir uma maior utilização dos enlaces e um melhor compartilhamento entre as restrições de banda definidas, considerando que as classes de tráfego inferiores podem utilizar a banda não utilizada nas classes hierarquicamente superiores e vice-versa. Outro aspecto observado é o fato dele preservar as garantias mínimas de banda previstas nos acordos de serviços e, além disso, propiciar uma eventual melhoria da qualidade do atendimento do mesmo em determinados cenários de distribuição de tráfego.

Shnayder e outros (2014) projetaram um protocolo para a priorização dinâmica de dados em roteadores com acesso compartilhado, como dispositivos 3G e 4G. O mecanismo prioriza a largura de banda em favor dos usuários com o maior valor. Um mecanismo de *pool* foi adotado como parte do protocolo, a fim de maximizar o uso de métodos de priorização de tráfego. Outro fator de sucesso é a identificação de classes de modelos de demanda estocástica e um esquema de priorização que forneça de forma eficiente a alocação de banda. Os resultados da simulação confirmam ganhos de eficiência a partir da priorização dinâmica em relação aos métodos tradicionais, bem como a eficácia do uso de *pool*.

Suliman e outros (2004), propuseram um modelo que aborda o problema da alocação de recursos em redes heterogêneas sem fio. Neste modelo, as redes sem fio devem cooperar umas com as outras para atender às solicitações de alocação de recursos através da formação de coalizões entre si. O modelo proposto permite que um usuário móvel possa dividir a sua aplicação entre os membros da coalizão, quando as aplicações não puderem ser manuseadas por uma única rede. Os autores utilizaram técnicas de balanceamento de carga com uma função de custo exponencial para equilibrar a carga nas redes e aplicaram os conceitos de teoria dos jogos cooperativos em redes de computadores (A. B. MacKenzie, 2001). A principal contribuição deste trabalho, segundo os autores, foi a aplicação do conceito de jogos cooperativos para resolver o problema de alocação de recursos em redes heterogêneas sem fio. Foi apresentada uma estrutura de coalizão, que permite múltiplas redes cooperarem para cumprir exigências de alocação de recursos, maximizando o resultado pretendido e satisfazendo os requisitos de desempenho do usuário. Essa mesma estratégia de alocação de recurso, utilizando a teoria dos jogos cooperativos com zonas de coalizão de interesse comum, também foi utilizada em outros trabalhos (ANTONIOU; PITSILLIDES, 2007; JAHAN; HASSAN; DAS, 2010; KHAN et al., 2008; KHAN et al., 2009; KHAN et al., 2011; MANDRU; BAVIRISETTI; KHARA, 2012; SHIN et al., 2012; TRESTIAN; ORMOND; MUNTEAN, 2011). É apresentada em Suliman et al. (2004) uma estratégia que utiliza a teoria dos jogos na formação de grupos de coalizão, objetivando um gerenciamento de interesse comum, a fim de detectar quais redes estão menos sobrecarregadas e distribuir os fluxos entre elas, dando a percepção que os fluxos estão em uma única rede.

Salgueiro (2009) foi fundamental para que o modelo EXP-RULE-SH fosse proposto

por Iturralde et al. (2011), sendo adaptado para a alocação de recursos em redes (*Long Term Evolution* - LTE[5]). Este modelo é um algoritmo de escalonamento de pacotes baseado na junção do algoritmo EXP-RULE P. Ameigeiras, J. Wigard (2004), com a aplicação do Valor de *Shapley*, associando a Teoria dos Jogos Cooperativos combinado com jogos de falência. O valor de *Shapley* é um conceito proposto em Teoria dos Jogos por *Lloyd Shapley* L. S. Shapley (1953) com o objetivo de propor uma repartição justa dos lucros obtidos coletivamente entre os vários jogadores. O valor de *Shapley* foi amplamente discutido em Salgueiro (2009) como sendo um método para divisão equitativa. Em um jogo cooperativo, grupos de jogadores, ou "coalizões", podem impor o comportamento cooperativo. Portanto, o jogo é uma competição entre as coalizões de jogadores, ao invés de entre os jogadores individuais. A análise de situações de falência na teoria dos jogos cooperativos tenta prescrever como racionar um montante de recursos perfeitamente divisíveis entre um grupo de jogadores de acordo com um perfil de exigências que, no conjunto, excede a quantidade a ser distribuída (SALGUEIRO, 2009). Para executar o modelo de alocação de recursos proposto, foi definido um ambiente de simulação. Foi usada uma única célula com interferência, quando há 40% de usuários que utilizam fluxos de vídeo, 40% com base em fluxos de voz sobre IP (*Voice Over Internet Protocol* - VoIP) e os restantes 20% estão usando fluxos CBR (*Constant Bit Rate*). Os usuários estão constantemente se movendo a uma velocidade de 3 milhas por hora em direções aleatórias. O simulador utilizado para executar todo o processo foi o LTE-Sim (G. Piro, L. Grieco, G. Boggia, F. Capozzi, 2010). Os resultados obtidos demonstraram um melhor desempenho do algoritmo EXP-RULE-SH proposto neste trabalho quando comparado com os demais "PF, M-LWDF e EXP-RULE"citados em (R. Basukala, H. Mohd Ramli, 2009). Os resultados mostram que, um mecanismo de distribuição de recursos realizados por valor de *Shapley* em jogos de utilidade de transferência, usado em conjunto com um algoritmo de escalonamento inteligente como EXP-RULE, pode melhorar a eficiência deste algoritmo entre 25% a 30%.

O modelo de restrição MAX-MIN proposto por Shan, Yang e Member (2007), foi projetado para gerir a largura de banda e apoiar a engenharia de tráfego em ambiente com serviços diferenciados (*Differentiated Services* - DiffServ). Os autores destacam algumas contribuições neste artigo. Em primeiro lugar, a realização de um gerenciamento de banda em duas dimensões com DS-TE (*DiffServ-Aware Traffic Engineering*): CT (*Class Type*) e prioridade. Ela permite utilizar matrizes na gestão de largura de banda. Em segundo lugar, foi proposto um novo modelo de restrição de banda, em que ambos os mínimos e máximos garantidos são configurados para cada tipo de classe CT. A ideia chave é um dilema "usá-lo ou emprestá-lo", o que garante uma largura de banda mínima para cada CT sem causar fragmentação e desperdício e, em terceiro lugar, três novos algoritmos de preempção de largura de banda são apresentados para os modelos MAM, RDM (*Russian Doll Model*), e o modelo de restrição proposto MAX-MIN, respectivamente. O MAX-MIN objetiva à realização de justiça na preempção, robustez de

[5] LTE é um padrão para comunicação sem fio de alta velocidade para telefones celulares e terminais de dados, com base nas tecnologias GSM / EDGE e UMTS / HSPA

desempenho e também melhorar a eficiência da largura de banda. Tanto o MAM quanto o RDM não possuem garantias de largura de banda mínimas para CTs, assim, não podem garantir a taxa de serviço dos aplicativos e que requisitos de QoS sejam independentes da intensidade das classes de tráfego. Para resolver este problema, os autores propuseram no modelo que tanto o mínimo garantido e as máximas restrições de largura de banda reservável sejam configurados para cada CT.

3.2 Redes definidas por software

O sistema *CAching in Buckets* - CAB proposto por Yan et al. (2014), utiliza uma estratégia baseada em *cache* combinada com uso de regras associativas que são aplicadas em SDN. A ideia principal consiste no particionamento lógico de espaços em estruturas denominadas de campos lógicos em *caches*, juntamente com todas as regras de associação. Os autores afirmam que o CAB promove e resolve o problema de dependências de associação de regras com baixa sobrecarga de armazenamento. Quando comparado com outros modelos semelhantes, o sistema proposto reduz significativamente os pedidos de fluxos de configuração, economizando e controlando a largura de banda, bem como reduz o tempo médio dos fluxos de configuração, promovendo assim um uso mais eficiente dos recursos nas SDN.

Em seu trabalho, Chandrasekaran e Benson (2014), apresentara pontos sobre vulnerabilidades em SDN, que segundos os autores dificultam a sua adoção, apesar dos benefícios já comprovados. As vulnerabilidades destacadas são confiabilidade e tolerância a falhas. Contrapondo esses itens, os autores propuseram um novo *design* de arquitetura SDN, cujo objetivo é a descentralização dos controladores, tornando-os assim mais tolerantes a falhas. Tal modelo de arquitetura, segundo os autores, melhora a confiabilidade de um ambiente SDN e elimina as possíveis resistências para sua adoção.

Foi proposto por AuYoung e outros (2014) um mecanismo automático de resolução baseado em uma família de procedimentos de votação aplicado para resolver conflitos de recursos entre SDN e programas de controle em nuvem. Foi observado que a escolha adequada de um mecanismo de resolução depende de duas propriedades implementadas em módulos: a primeira diz respeito a precisão e a segunda a paridade. Com base nessas propriedades, um operador de rede poderá aplicar uma infinidade de técnicas de resolução. Esse mecanismo, segundo os autores, promove a modularidade e não requer que cada módulo controlador divulgue seus objetivos ou algoritmos para os demais módulos. Esta técnica apresentou uma melhoria na qualidade da alocação quando comparado com outros métodos de resolução, como atribuição de prioridades estáticas ou por igual peso e decisões via *round-robin*. Também foi apresentado uma comparação qualitativa deste mecanismo com métodos mais recentes baseados em utilidade ou valor.

Katiyar e outros (2015) propuseram um mecanismo de autoconfiguração para novos *switches* em SDN, redes tradicionais e em redes SDN híbridas. A automação das configurações

iniciais dos *switches* SDN, trazem vantagem e reduzem os custos de instalação e manutenção em redes corporativas. No entanto, a heterogeneidade de uma rede híbrida apresentam dificuldades na obtenção de tais benefícios. O sistema proposto possui as seguintes características: (i) a detecção de um novo *switch* SDN em uma rede híbrida; (ii) configura o *switch* com parâmetros iniciais de configuração em rede SDN híbrida e não híbrida e; (iii) assegura um serviço contínuo através para redes SDN e não SDN, garantindo uma maior manutenibilidade gerencial da infraestrutura de rede e abstraindo toda sua complexidade.

Jogos em nuvem (*Nuvem Gaming*) é um serviço emergente que começou a ganhar destaque na indústria de jogos *online* (AMIRI et al., 2015). O processamento, a compressão e renderização de vídeo são realizados em *data centers* dedicados, necessitando um alto grau de controle dos fluxos de dados dentro da nuvem. Estes elementos influenciam diretamente sobre a qualidade dos serviços. Neste contexto, Amiri e outros (2015), apresentaram duas contribuições: foi criado um projeto para aplicar o paradigma SDN ao contexto de *Nuvem Gaming*. Para tal, foi desenvolvido um controlador SDN que reduz os atrasos e as variações captadas pelos jogadores. Este controlador SDN, dispersa a carga de tráfego do jogo de forma adaptativa entre os diferentes caminhos de rede de acordo com as variações e atrasos. Os resultados experimentais mostram que o controlador proposto reduz o atraso e a variação do atraso fim-a-fim em quase 9% e 50% respectivamente, sem gerar perdas de pacotes adicionais, quando comparado ao método convencionai: *Open Shortest Path First* - OSPF. Estes ganhos promovem uma melhor experiência aos jogadores no tocante aos jogos.

3.3 Compartilhamento de banda em SDN

Foi proposto por Tian e outros (2016) um sistema de alocação de largura de banda baseado em SDN, *FairShare*, que eficientemente oferece compartilhamento justo de largura de banda. Em SDN é possível obter informações sobre o estado da rede em tempo real, gerenciar largura de banda, controlar fluxos de dados em *switches* OpenFlow e implantar dinamicamente novas políticas de QoS ou alterá-las. O *FairShare* adota o algoritmo de Equivalência Max-min (LMF) Local-link para alocar a largura de banda. O LMF é projetado como um algoritmo distribuído, que é escalável e eficiente para alocação de largura de banda. Foi implementado um protótipo na plataforma Mininet e um sistema de simulação, o experimento mostra que a *FairShare* pode oferecer rapidamente uma alocação de banda justa, uma vez que a rede se torna injusta.

Com a evolução do paradigma de rede definida por *software* (SDN), o gerenciamento de tráfego em redes de data center tornou-se flexível e escalável. A solução existente usando o OpenFlow, o mecanismo de garantia de taxa, é ineficiente, pois limita a taxa dos fluxos ao retirar fluxos de pacotes para alcançar a taxa de transferência desejada. Neste contexto, foi proposto por Mohan e outros (2016) um algoritmo chamado de BASIS, uma solução baseada na inferência

bayesiana para fornecer garantias proporcionais de Qualidade de Serviço (QoS) aos serviços em uma rede de *datacenter*. Com o BASIS, a largura de banda de um link congestionado de saída será compartilhada entre os fluxos competitivos em proporção aos pesos escolhidos por eles. É usado a inferência bayesiana para capturar a história das taxas de chegada do fluxo e a carga oferecida usando uma única fila e estimar a probabilidade de queda diferencial de fluxos de maneira que respeite os pesos atribuídos a eles na chegada. Ao contrário da abordagem de limitação de taxa, a BASIS descarta proativamente um pacote de um fluxo de forma probabilística para alcançar o rendimento desejado e evita a perda de fluxos de pacotes. A solução proposta é avaliada em uma plataforma SDN emulada e demonstra que a BASIS atinge a taxa de transferência desejada com menor número de pacotes que as abordagens existentes.

O problema de equidade entre sessões *multicast* é devido ao compartilhamento de recursos de rede entre um conjunto de sessões de *multicast*. Além da equidade na alocação de recursos, o ajuste mínimo de recursos e a menor complexidade computacional são outros dois requisitos importantes em redes reais. Neste sentido, (Tang e outros (2015)) realizaram um estudo e propuseram um algoritmo heurístico para resolver o problema no contexto das redes SDN para fornecer canais de transmissão de vídeo codificados. Comparado com os algoritmos existentes, o algoritmo heurístico possui um maior equilíbrio da alocação de recursos usando o mínimo de ajustes de recursos, além de possuir a menor complexidade computacional.

As redes acadêmicas sem fio atuais representam um desafio de gerenciamento de acesso ao usuário. Ou seja, uma rede acadêmica deve prestar serviços a diversos dispositivos de usuários cujas políticas de alocação de recursos devem ser tratadas de forma diferente de acordo com a política da universidade. A rede definida por *software* (SDN) é um candidato adequado para enfrentar o desafio devido aos seus recursos de controle centralizado e dinâmico. Com base no protocolo OpenFlow, que é um premissa para realizar uma rede definida pelo *software*, os autores Huang e outros (2014) implementaram uma técnica de corte de largura de banda em um AP WiFi. O sistema operacional no AP é exibido no sistema OpenWrt Linux para habilitar a função de cliente OpenFlow. Resultados experimentais mostram que usuários em diferentes grupos experimentam diferentes quantidades de largura de banda e as proporções de compartilhamento da largura de banda sem fio podem ser ajustadas dinamicamente em tempo real. Além disso, resultados mostram qualidade de vídeo diferente em dispositivos de usuários de diferentes grupos, verificando a aplicabilidade da técnica de corte de largura de banda.

3.4 Síntese

A Tabela 1 apresenta os principais conceitos chave desta dissertação. Os artigos apresentados neste capítulo, serviram como arcabouço intelectual e indicaram possíveis caminhos de pesquisa. É importante ressaltar que, as buscas foram realizadas nas bases mencionadas no período de 03/2013 a 08/2017 e não foi encontrado qualquer trabalho que possua todos os

conceitos expostos na Tabela 1, caracterizando a singularidade da nossa pesquisa.

Tabela 1 – Trabalhos Relacionados

Num	Artigo	SDN	Multisserviços	QoS	OpenFlow	Compartilha Banda	Restrição de Banda
01	Reale e outros (2011)		x	x		x	x
02	Shnayder e outros (2014)		x	x		x	
03	Suliman e outros (2004)		x	x			
04	Salgueiro (2009)		x	x			
05	Shan e outro (2007)		x	x		x	x
06	Yan e outros (2014)	x				x	
07	Chandrasekaran e outros (2014)	x					
08	AuYoung e outros (2014)	x					
09	Katiyar e outros (2015)	x	x				
10	Amiri e outros (2015)	x		x			
11	Tian e outros (2016)	x		x	x	x	
12	Mohan e outros (2016)	x		x	x		
13	Tang e outros (2015)	x		x			
14	Huang e outros (2014)	x		x	x	x	
15	Carvalho (2017)	x	x	x	x	x	x

4

Serviço de Compartilhamento de Banda

Este capítulo apresenta o serviço de compartilhamento de banda em redes definidas por *software* usado nesta dissertação. Inicialmente é abordada a seção 4.1 Arquitetura do Serviço BSS/SDN; na seção 4.2 a Aplicação Desenvolvida e na seção 4.3 o modelo de barganha implementado.

4.1 Arquitetura do Serviço BSS/SDN

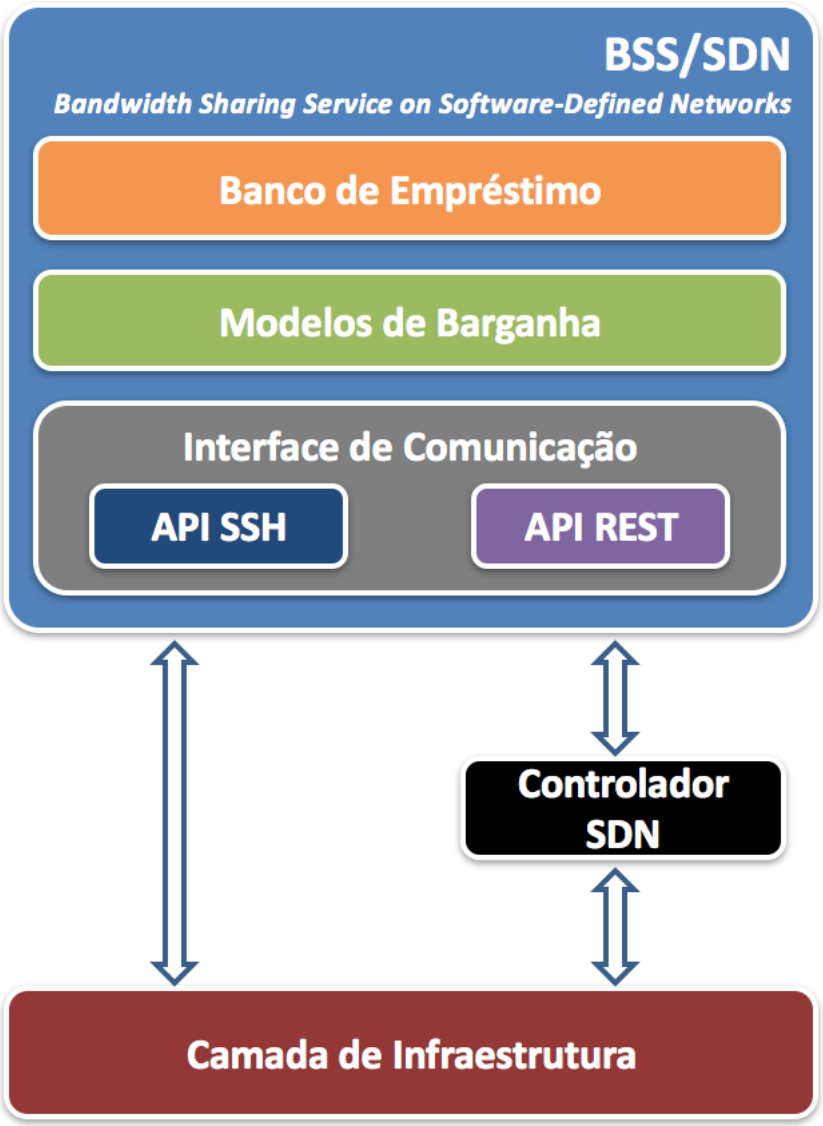
Inspirado pelo mecanismo de barganha definido em Salgueiro (2009), foi proposta a arquitetura de serviço BSS/SDN (*Bandwidth Sharing Service on Software-Defined Networks*) definida em camadas. Esta arquitetura de serviço é apresentada na Figura 6, tendo em vista as características de extensibilidade para novos modelos de barganha, uso de banco de empréstimo e possui uma interface de comunicação com duas APIs.

As camadas são definidas nas subseções: 4.1.1 camada do banco de empréstimo; 4.1.2 camada dos modelos de barganha; 4.1.3 camada de comunicação; 4.1.4 camada do controlador SDN e 4.1.5 camada de infraestrutura de rede.

4.1.1 Banco de Empréstimo

Camada responsável por promover a persistência dos dados provenientes dos modelos de barganha em banco de dados. Estes dados podem ser estatísticos dos dispositivos OpenFlow oriundos do controlador SDN ou dados de negócio pertencentes ao mecanismo de barganha, tais como: empréstimos, tamanho das filas, dados de configuração e etc.

Figura 6 – Arquitetura de serviço BSS/SDN.



4.1.2 Modelos de Barganha

Esta camada foi definida a fim de possibilitar a agregação de novos modelos de barganha. Entende-se por modelo de barganha, toda à regra de negócio que determine seu mecanismo de funcionamento, ou seja, como os empréstimos deverão ser realizados e em quais circunstâncias eles acontecerão. Salgueiro (2009) em sua pesquisa, abordou o valor de *shapley* como sendo um mecanismo de justiça para a alocação de banda em um cenário competitivo. Abrahao (2015) propôs dois esquemas de alocação de blocos de recurso para a transmissão de *downlink* LTE. O primeiro esquema proposto utiliza o critério Max-min e o segundo emprega um sistema de inferência fuzzy para calcular as prioridades dos usuários e tomar decisões de escalonamento e Xavier (2012) utiliza técnicas como lógica fuzzy, algoritmos genéticos e sistemas multiagentes com o objetivo de alocar dinamicamente tarefas a recursos.

4.1.3 Interface de Comunicação

Camada responsável por prover comunicação de rede TCP/IP. Esta camada oferece duas APIs de comunicação. A primeira delas é a API SSH (*Secure SHell*), que tem como característica permitir uma comunicação segura, possibilitando realizar configurações contingenciais e emergenciais em dispositivos OpenFlow localizados na camada de infraestrutura. Já a API REST (*Representational State Transfer*), é responsável por servir como base para o desenvolvimento de aplicações na Web. Similarmente ao *web services*, um serviço REST é independente de plataforma ou linguagem de programação. O protocolo HTTP é usado para fornecer as características necessárias da comunicação (FIELDING, 2000).

4.1.4 Controlador SDN

O controlador oferece um plano de controle dinâmico xpara automatizar e programar a rede visando torná-la mais ágil. É a parte principal do paradigma SDN que funciona entre dispositivos de rede (camada de infraestrutura) e as diversas aplicações que fornecem uma interface extensível de programação para rede. Um controlador SDN localiza-se na camada de controle e possui a responsabilidade de gerenciar os protocolos de rede, as políticas e estabelecer o caminho na rede (MORZHOV; ALEKSEEV; NIKITINSKIY, 2016). A comunicação entre o controlador e a camada de infraestrutura de rede se dá através do protocolo OpenFlow.

4.1.5 Infraestrutura de Rede

Nesta camada concentram-se os diversos dispositivos OpenFlow, tais como: roteadores, *switches*, *switches* virtuais, *access point* e demais dispositivos que possuam suporte ao protocolo OpenFlow, independentes se são virtuais ou físicos.

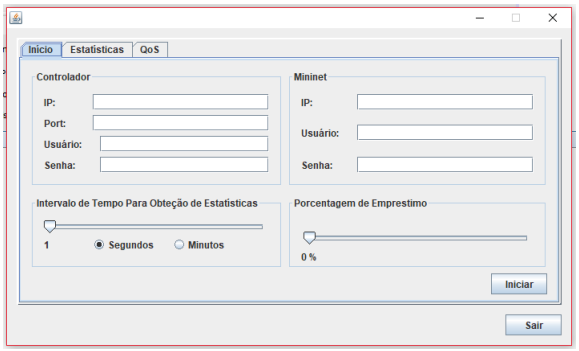
4.2 Aplicação Desenvolvida

Esta seção apresenta a aplicação de compartilhamento de banda em redes SDN desenvolvida. Através dela é possível realizar parametrizações e configurações que os experimentos demandam, bem como oculta do usuário toda complexidade sistêmica foi implementada usando a linguagem de programação JAVA (ORACLE, 2017), pois trata-se de uma linguagem orientada a objetos, de fácil implementação e compreensão, como também é a mesma linguagem adotada pelo controlador SDN FloodLight adotado neste trabalho. Associada a esta aplicação, foi projetado um banco de dados com o objetivo de acomodar os dados, tais como: estatísticas dos dispositivos de rede e regras de negócio essenciais para a interpretação dos resultados.

4.2.1 Interface Gráfica do Usuário

Nesta subseção apresentam-se as interfaces gráficas dos usuário GUI (*Graphical User Interface*). A Figura 7 apresenta a GUI inicial e com ela é possível realizar as parametrizações essenciais , tais como:.

Figura 7 – Tela Inicial de Configuração.

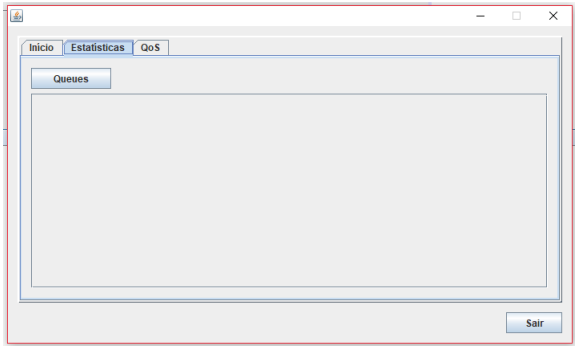


- **Controlador:** Parâmetros de endereçamento e autenticação para com o controlador SDN;
- **Mininet:** Parâmetros de endereçamento e autenticação para com o emulador de rede Mininet;
- **Intervalo de Tempo:** Define de quanto em quanto tempo (segundos ou minutos) que a aplicação solicitará ao controlador as estatísticas dos dispositivos OpenFlow;
- **Percentual de Empréstimo:** Percentual a ser usado como taxa de empréstimo (crédito ou débito). Esta taxa será usada como parâmetro de entrada para o Algoritmo 1 no momento dos empréstimos.

Com as configurações acima definidas, clica-se no botão de comando [Iniciar] para deixar a Aplicação funcional.

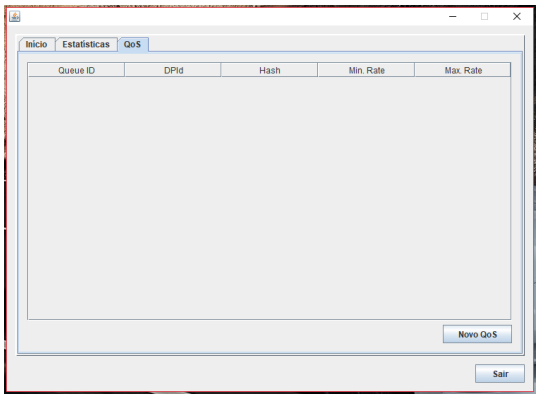
A Figura 8 apresenta a GUI de estatísticas. Através dela é possível visualizar uma lista das estatísticas das *queues* contidas em cada *switch* OpenFlow.

Figura 8 – Tela de Configuração das Filas.



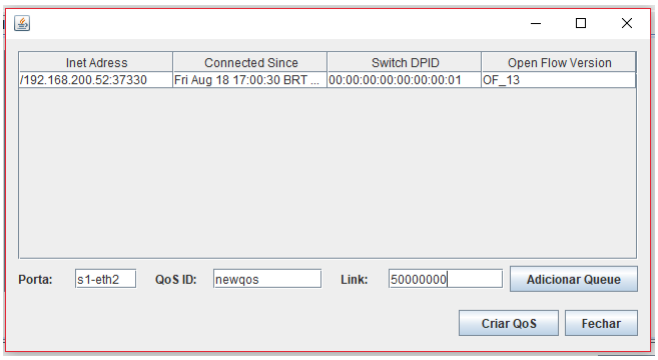
A Figura 9 exibe a GUI de QoS. Aqui são listadas as configurações de QoS, tais como: tamanho mínimo e máximo de cada fila. Também é possível adicionais novas políticas de QoS através do botão de comando [Novo QoS] localizado na parte inferior à direita da GUI.

Figura 9 – Tela de Configuração Inicial de QoS.



A Figura 10 ilustra a GUI do botão de comando [Novo QoS]. Nesta é possível ver os dados de configuração da QoS, possibilitando também a adição de novas filas associadas a QoS em questão. A parte inferior da GUI contém as caixas de entradas de parâmetros: porta, QoS id, *oink* e o botão de comando [Adicionar *Queue*].

Figura 10 – Tela de Criação de QoS.



A Figura 11 apresenta a GUI do botão de comando [Adicionar *Queue*]. Nesta GUI são definidos os parâmetros de configuração das *queues*, tais como: id, tamanho mínimo e máximo.

4.2.2 Diagrama Entidade-Relacionamento

É apresentada nesta subseção o Diagrama Entidade-Relacionamento (DER) concebido para a aplicação. Este DER possibilita a persistência dos dados estatísticas e/ou de negócios, tais como: dados dos *switches*, das portas, das filas, bem como os fluxos contidos nos *switches* e tabelas de fluxos.

Figura 11 – Tela de Criação das Filas.

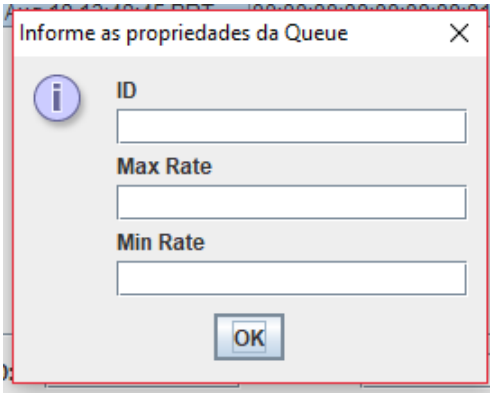
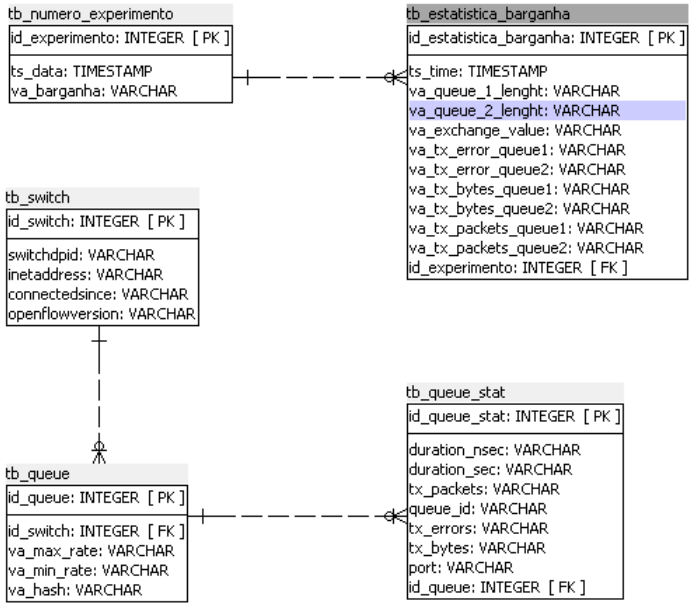


Figura 12 – Diagrama de Entidade de Relacionamento.



4.3 Modelo de Barganha Implementado

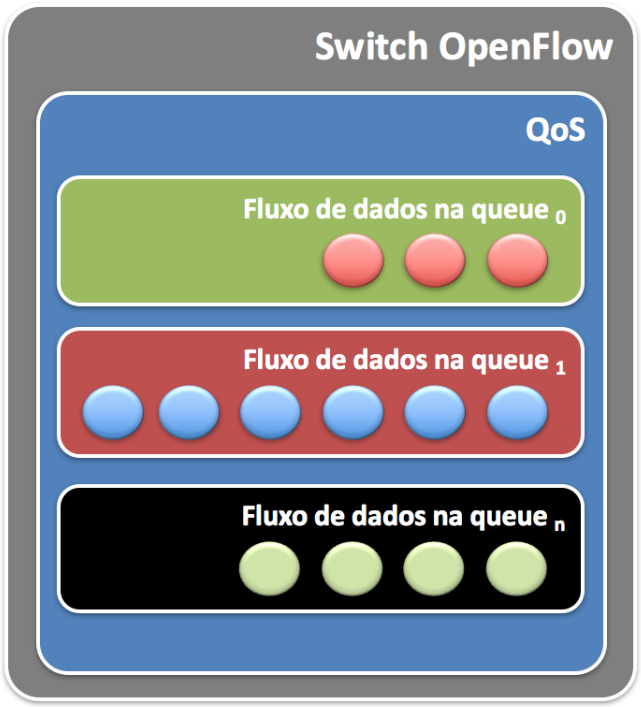
O problema da barganha ou da negociação resume-se na seguinte ideia: existe fundamento para uma negociação quando pelo menos dois agentes[1] têm a possibilidade de aumentar seu ganho caso cheguem a um acordo entre si (NASH, 1950).

Neste contexto, a fim de otimizar a taxa nominal máxima da largura de banda em redes SDN por meio do compartilhamento da banda, foi implementado um modelo simples de barganha,

[1] Agentes são caracterizados neste trabalho como sendo filas de QoS.

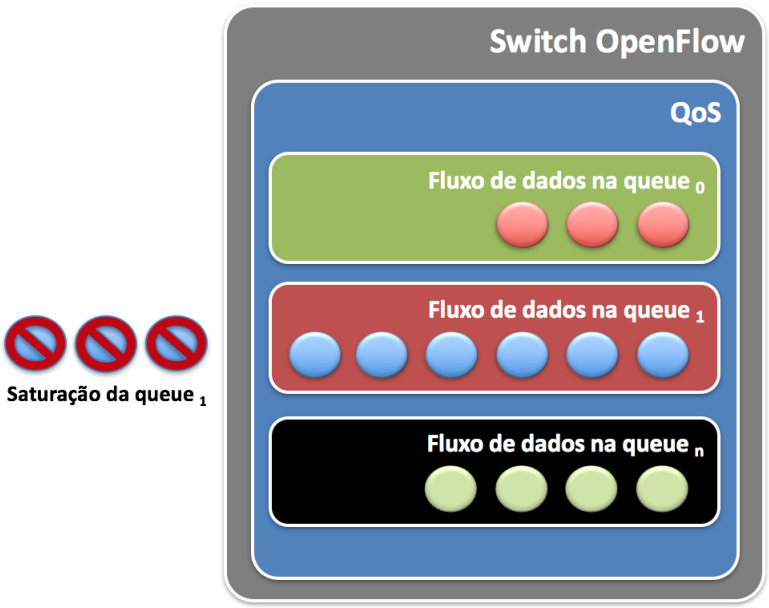
que demonstra a aplicabilidade funcional da arquitetura BSS/SDN. O compartilhamento de banda é realizado nas *queues* ativas definidas em uma política de QoS do *switch* OpenFlow como demonstrada na Figura 13.

Figura 13 – Switch OpenFlow com QoS e duas Queues.



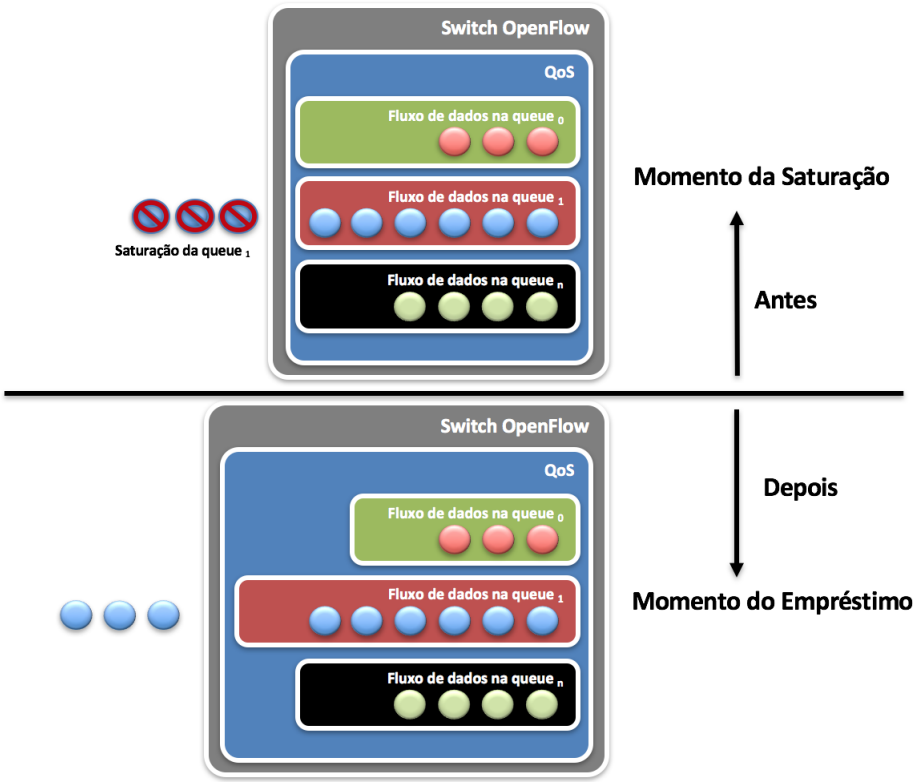
Este compartilhamento dar-se por meio do monitoramento ativo das *queues*, a fim de detectar o momento exato de uma possível saturação como demonstrada na Figura 14.

Figura 14 – Saturação das Queues no switch OpenFlow.



Nesse momento, dar-se início ao jogo de empréstimos de banda não utilizada pelas *queues* que não sofreram saturação. Os pedidos usam como fator de empréstimo um percentual definido via aplicação como apresentado na GUI da Figura 7 (Configuração Inicial), onde a *queue* que estiver precisando de mais banda, solicitará este percentual, a fim de minimizar sua perda de pacotes. A Figura 15 demostra esse mecanismo.

Figura 15 – Empréstimo entre as Queues no switch OpenFlow.



Neste contexto, o jogo de empréstimos é orquestrado pelo mecanismo de barganha, que por sua vez fará uso do modelo de barganha exemplificado pelos algoritmos 1 (Monitoramento das Filas) e 2 (Algoritmo de Barganha). É importante ressaltar que, a separação dos fluxos em *queues* é pré-requisito para aplicar o compartilhamento de banda proposto, uma vez que os empréstimos são realizados sobre elas.

O Algoritmo 1 possui como entrada de dados as n *queues* que participam dos empréstimos e o percentual de empréstimo definido na GUI da Figura 7 (Configuração inicial). Seu mecanismo lógico de funcionamento é o seguinte: (1) a cada fração de tempo (percentual de empréstimo) obtido faça; (2) obtém as estatísticas das *queue* em banco de dados; (3) verifica se há saturação em alguma *queue*; (4) executa o algoritmo de barganha passando as *queues* participantes do jogo de empréstimos e a identificação da *queue* saturada.

O Algoritmo 2 possui como entrada de dados as n *queues* que participam dos empréstimos e a identificação da *queue* que está solicitando o empréstimo. Seu mecanismo lógico de funcionamento para os casos de empréstimos sem uso do banco de empréstimos: (1) verifica se há disponibilidade de realizar empréstimos; (2) sendo possível, obtém a taxa de empréstimo;

Algoritmo 1: Monitoramento das Filas

```
   Entrada: vFilas[], PercTempo
   Saída: Boolean
1  início
2      inicialização;
3      while PercTempo do
4          ObtemEstatisticas(vFilas[]);
5          if ExistiSaturacaoNasFilas(vFilas[]) then
6              IdFilaSolEmp ← ObtemFilaSaturada(vFilas[]);
7              AlgoritmoDeBarganha(vFilas[], IdFilaSolEmp);
8      end
9  fim
```

(3) identifica a *queue* que possui a disponibilidade de conceder o empréstimo; (4) efetiva o empréstimo passando as informações obtidas anteriormente (as *queues* participantes, a *queue* que empresta, a *queue* que solicitou o empréstimo e a taxa de empréstimo) e; (5) atualiza o banco de empréstimos com as informações da negociação.

Caso não haja disponibilidade de empréstimos sem consulta ao banco de empréstimo, o fluxo será o seguinte: (1) verifica se existe algum crédito a favor da *queue* solicitante junto as *queues* que participam dos empréstimos; (2) existindo ha possibilidade de se obter o crédito devido, obtêm-se a taxa de empréstimo; (3) identifica a *queue* que possui a dívida no banco de empréstimos; (4) efetiva o empréstimo passando as informações obtidas anteriormente (as *queues* participantes, a *queue* que empresta, a *queue* que solicitou o empréstimo e a taxa de empréstimo) e; (5) atualiza o banco de empréstimos com as informações da negociação.

Caso nenhuma das duas formas de empréstimos de banda for possível, a fila solicitante descartará seus pacotes e o jogo continua.

Algoritmo 2: Algoritmo de Barganha

Entrada: vFilas[], IdFilaSolEmp

Saída: Boolean

```
1  início
2      inicialização;
3      if haDisponibilidadeDeEmprestimo(vFilas[]) then
4          TxEmp ← ObtemTxEmp();
5          IdFilaQueEmpresta ← DisponibilidadeDeEmprestimo(vFilas[]);
6          EfetuaEmp(vFilas[], IdFilaQueEmpresta, IdFilaSolEmp, TxEmp);
7          AtualizaBancoDeEmprestimo();
8      else
9          if AlguemMeDeve(BancoDeEmp(), IdFilaSolEmp) then
10             TxEmp ← ObtemTxEmp();
11             IdFilaMeDeve ← QuemMeDeve(BancoDeEmp(), IdFilaSolEmp);
12             EfetuaEmp(vFilas[], IdFilaMeDeve, IdFilaSolEmp, TxEmp);
13             AtualizaBancoDeEmprestimo();
14         else
15             NaoRealizaEmprestimo();
16         end
17     end
18 fim
```

5
Estudos de Casos

Neste capítulo são apresentas as seguintes seções: A seção 5.1 descreve os experimentos e os ambientes de validação e de estudo de caso, os recursos utilizados em cada experimento e os cenários de execução. Na seção 5.2 é descrita toda a metodologia de avaliação utilizada nesta parte do trabalho, abrangendo as métricas usadas e a definição da carga de trabalho. A seção 5.3, apresenta uma validação do uso do serviço de compartilhamento de banda num cenário simples de validação ao fazer uso dos protocolos TCP e UDP, bem como apresenta uma análise comparativa dos resultados e por fim, a seção 5.4, apresenta um estudo de caso onde demonstra num cenário competitivo por banda o serviço de compartilhamento de banda ora proposto neste trabalho. Ainda nesta seção é apresentada uma análise comparativa dos resultados.

5.1 Experimentos

Para realizar os experimentos, foi adotada como base a topologia usada por GOMES et al. (2013), onde retrata uma topologia de rede típica SDN, como ilustrada na Figura 16. De modo geral, ela é composta por três máquinas interligadas através de um *switch*. Uma dessas máquinas atua como cliente, enviando requisições; outra atua como servidor, que responde as requisições dos clientes; e a terceira atuam como controlador SDN da rede. Para o estudo de caso, a mesma arquitetura foi estendida para cinco máquinas, sendo três delas atuando como cliente como ilustrada na Figura 17.

5.1.1 Ambientes

Embora a solução possa ser empregada em ambientes reais, nesta dissertação foi utilizado um ambiente emulado para caracterização dos elementos que compõem a topologia de rede. Para emular tais elementos, foi adotado o emulador de rede Mininet que é amplamente adotado pela comunidade científica a exemplo de (TIAN et al., 2016; JO et al., 2014; LANTZ; HELLER;

MCKEOWN, 2010). Foi utilizado o comutador virtual Open vSwitch, que oferece suporte a OpenFlow. É projetado para permitir a automatização de grandes redes através da extensão programática (MORALES; MURILLO; RUEDA, 2015; ZOPE; PAWAR; SAQUIB, 2016).

A topologia da rede adotada no experimento de validação é descrita a seguir, sendo exemplificada através da Figura 16 e da Tabela 2.

Figura 16 – Topologia da rede adotada no experimento de validação.

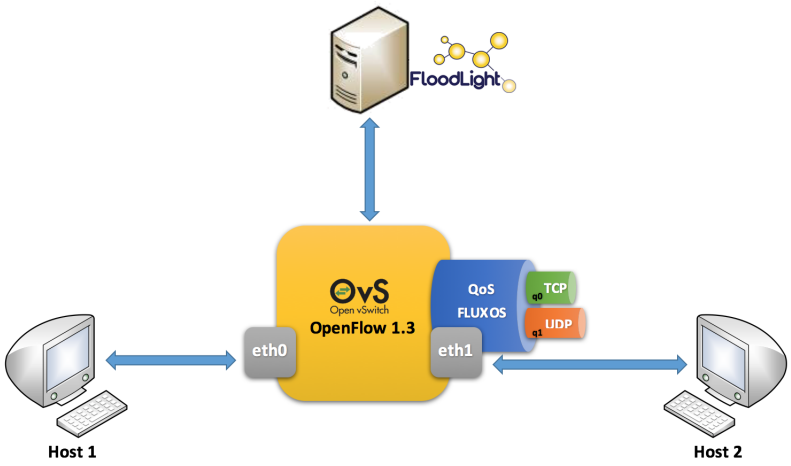


Tabela 2 – Detalhes da topologia da rede adotada no experimento de validação.

HOST	**TIPO**	**AMBIENTE**	**SISTEMA OPERACIONAL**
host 1	Virtual	Mininet	Ubuntu Linux
host 2	Virtual	Mininet	Ubuntu Linux
Comutador	Virtual	Mininet/Open vSwitch	Ubuntu Linux
Controlador SDN	Virtual	FloodLight	Ubuntu Linux

Apesar de que a solução possa ser empregada na classificação de fluxos de diversos protocolos entre **n** *queues* usando QoS, é exibido **n = 2** *queues*, conforme ambiente montado para o experimento. O tráfego de entrada das requisições **TCP** e **UDP** do *host* 1 chegam pela porta física **eth0** no *Open vSwitch* e é classificado entre as n *queues* na porta física de saída **eth1**. O controlador FloodLight é conectado ao *Open vSwitch*, possibilitando a comunicação entre os *hosts* presentes na rede através do protocolo OpenFlow.

Para a caracterização da topologia de rede no experimento do estudo de caso deste trabalho, temos a Figura 17 e a Tabela 3 que ilustra e detalha essa topologia.

Figura 17 – Topologia da rede adotada no experimento do estudo de caso.

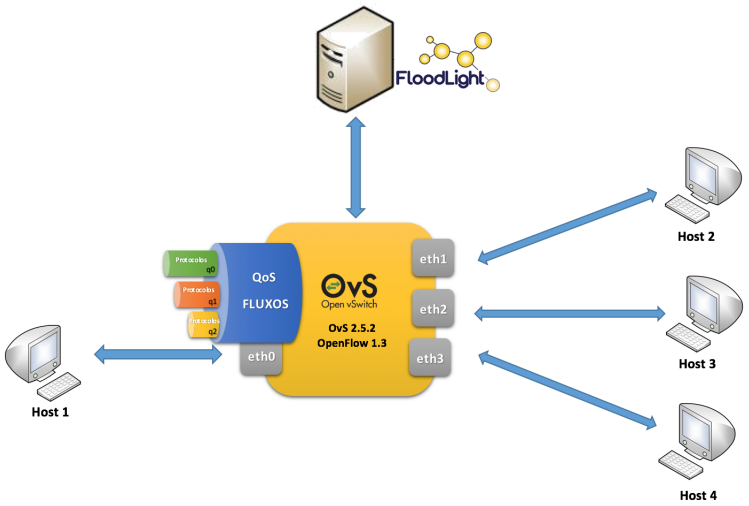


Tabela 3 – Detalhes da topologia da rede adotada no experimento do estudo de caso.

HOST	**TIPO**	**AMBIENTE**	**SISTEMA OPERACIONAL**
host 1	Virtual	Mininet	Ubuntu Linux
host 2	Virtual	Mininet	Ubuntu Linux
host 3	Virtual	Mininet	Ubuntu Linux
host 4	Virtual	Mininet	Ubuntu Linux
Comutador	Virtual	Mininet/Open vSwitch	Ubuntu Linux
Controlador SDN	Virtual	FloodLight	Ubuntu Linux

Pode-se observar que o tráfego de entrada das requisições oriundas do *host* 1 na porta física **eth0** são classificados através de regras de QoS baseadas em endereçamento **IP** e não mais por tipo de protocolo, sendo enviados para as respectivas *queues* do Open vSwitch para posteriormente serem encaminhados aos *hosts* finais aos quais estão diretamente conectados através das portas físicas de saída **eth1**, **eth2** e **eth3**. O controlador FloodLight é conectado ao *Open vSwitch*, a fim de possibilitar a comunicação entre os *hosts* presentes na rede através do protocolo OpenFlow. É importante ressaltar que, para esse estudo de caso o protocolo utilizado foi o UDP, mas poderiam ser utilizados diversos protocolos num cenários real, uma vez que não há restrições que impeçam tal uso.

5.1.2 Recursos Utilizados

As Tabelas 4 e 5 apresentam os recursos utilizados na execução dos experimentos de validação e do estudo de caso.

Tabela 4 – Recursos de *hardware*.

DESCRIÇÃO	**CPU**	**MEMÓRIA**	**SISTEMA OPERACIONAL**
Macbook Pro	2,6 GHz Intel Core i5	8 GB	macOS Sierra v10.12.6

Tabela 5 – Recursos de *software*.

DESCRIÇÃO	VERSÃO	TIPO
Ubuntu Desktop 64 bits	17.04	Sistema Operacional
FloodLight	1.2	Controlador SDN
Mininet	2.2.1	Emulador de Rede
Open vSwitch	2.5.2	Comutador de Pacotes
Parallels Desktop	12	*Software* de Virtualização

5.1.3 Cenários

A Figura 18 apresenta a topologia de rede usada nos cenários I e II do experimento de validação. Nesta topologia a comunicação entre os *hosts* h1 e h2 foi implementada através de QoS, utilizando o Open vSwitch para realizar a classificação do tráfego de acordo com o tipo de protocolo usado. Para o nosso experimento, foi considerado somente tráfego TCP e UDP. A caracterização em detalhes estão expostas na tabela 6. O objetivo deste cenário é validar o serviço de compartilhamento de banda em uma rede SDN.

Figura 18 – Topologia de rede usada nos cenários I e II.

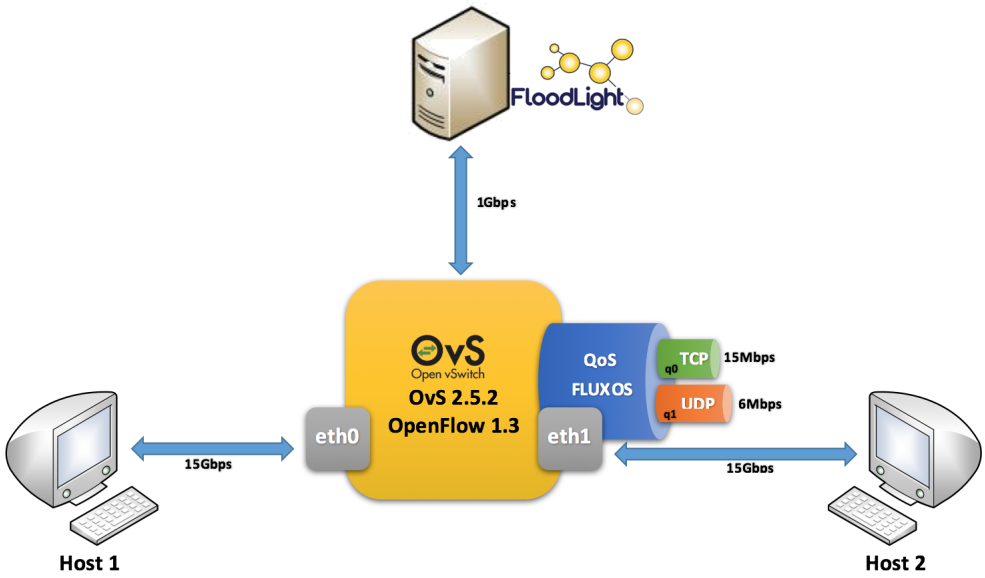


Tabela 6 – Cenários do Experimento de Validação.

CENÁRIO	DESCRIÇÃO
I	• Utiliza a mesma topologia de redes definida na Figura 18; • Faz uso de Qos para controle de fluxos; 1) Fila TCP: 15Mbps de capacidade; 2) Fila UDP: 6Mbps de capacidade. • Não faz uso do mecanismo de barganha; • Coleta as estatísticas para serem analisadas posteriormente.
II	• Utiliza a mesma topologia de redes definida na Figura 18; • Faz uso de Qos para controle de fluxos; 1) Fila TCP: 15Mbps de capacidade; 2) Fila UDP: 6Mbps de capacidade. • Faz uso do mecanismo de barganha como definida no capítulo 4 na seção 4.3; • Coleta as estatísticas para serem analisadas posteriormente.

As configurações de QoS especificadas na Tabela 6 foi aplicada e podem ser vistas em detalhes nas Figuras 19 e 20 abaixo.

Figura 19 – Detalhes da especificação das filas dentro da Aplicação - Experimento de Validação.

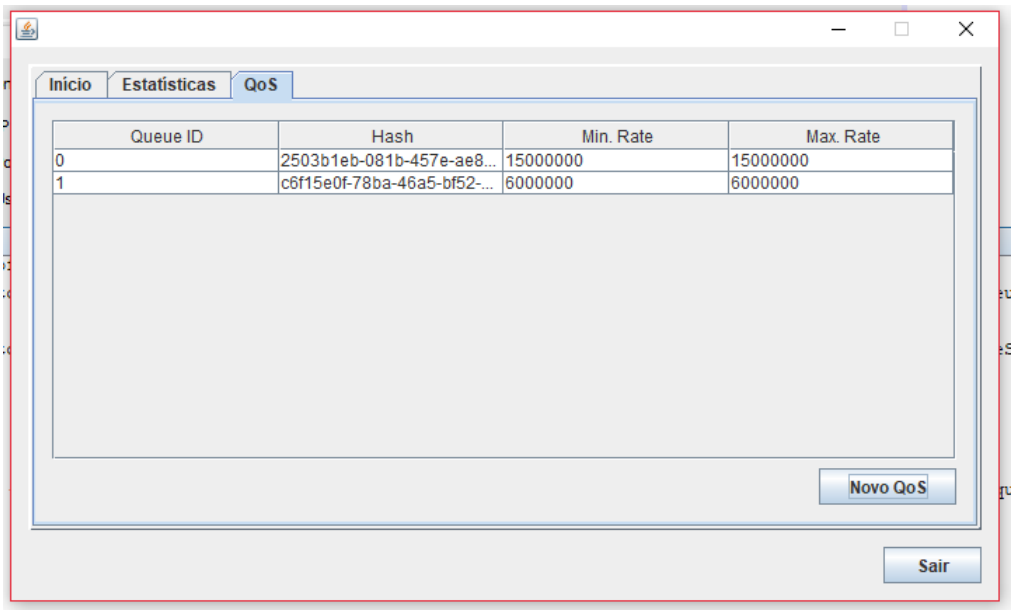


Figura 20 – Detalhes da especificação das filas dentro do Ovsdb - Experimento de Validação.

Já no experimento do estudo de caso, também temos dois cenários, mas submetidos em outra topologia como ilustrada na Figura 21. Nesta topologia, a comunicação entre todos os *hosts* foi implementada através de QoS, utilizando o Open vSwitch para realizar a classificação do tráfego de acordo com o endereçamento IP e não mais por tipo de procolo, pois este estudo de caso é independente do conteúdo trafegado. A caracterização em detalhes estão expostas na tabela 7. O objetivo deste cenário é demonstrar o serviço de compartilhamento de banda em uma rede SDN típica e independente do tipo de conteúdo trafegado.

Figura 21 – Topologia de rede usada nos cenários I e II.

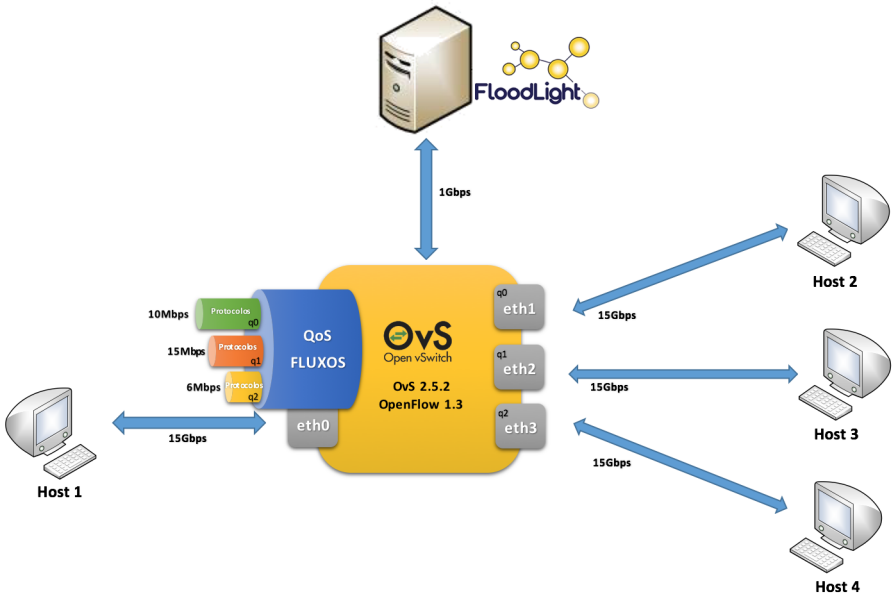


Tabela 7 – Cenários do Estudo de Caso.

CENÁRIO	**DESCRIÇÃO**
I	• Utiliza a mesma topologia de redes definida na Figura 21; • Faz uso de Qos para controle de fluxos baseados em endereçamento IP; 1) Fila 0: 10Mbps de capacidade; 2) Fila 1: 15Mbps de capacidade; 3) Fila 2: 6Mbps de capacidade. • Não faz uso do mecanismo de barganha; • Coleta as estatísticas para serem analisadas posteriormente.
II	• Utiliza a mesma topologia de redes definida na Figura 21; • Faz uso de Qos para controle de fluxos baseados em endereçamento IP; 1) Fila 0: 10Mbps de capacidade; 2) Fila 1: 15Mbps de capacidade; 3) Fila 2: 6Mbps de capacidade. • Faz uso do mecanismo de barganha como definida no capítulo 4 na seção 4.3; • Coleta as estatísticas para serem analisadas posteriormente.

As configurações de QoS especificadas na Tabela 7 foi aplicada e podem ser vistas em detalhes nas Figuras 22 e 23 abaixo.

Figura 22 – Detalhes da especificação das filas dentro da Aplicação - Experimento do Estudo de Caso.

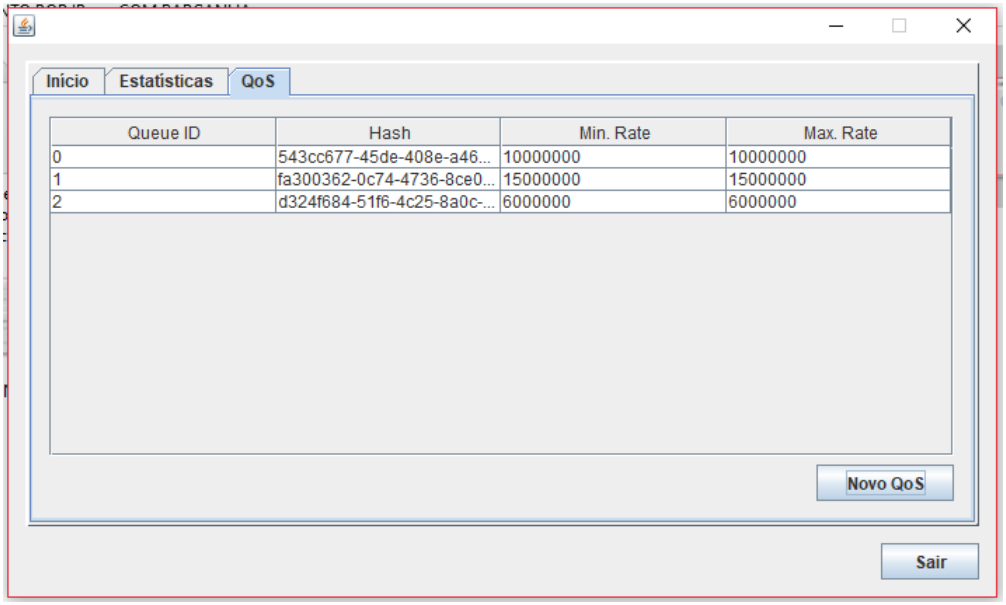
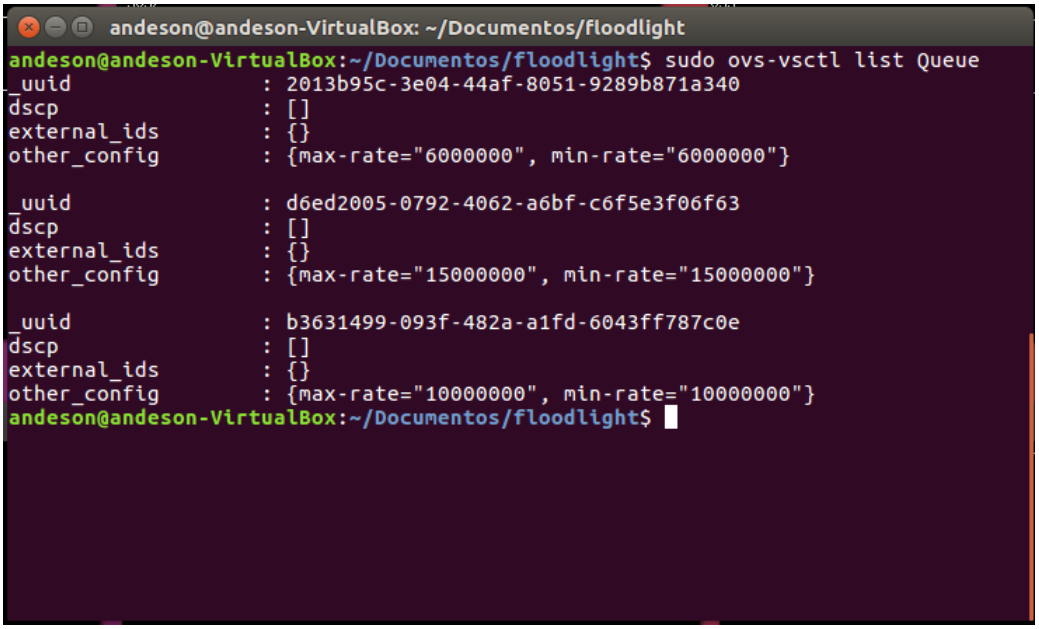


Figura 23 – Detalhes da especificação das filas dentro do Ovsdb - Experimento do Estudo de Caso.



5.2 Metodologia de Avaliação

Esta seção apresenta as métricas de desempenho adotadas, as quais dão subsídios para avaliar o desempenho do compartilhamento de banda em redes SDN bem como a carga de trabalho sugerida.

5.2.1 Métricas de Desempenho

As métricas de desempenho são os critérios pelos quais torna-se possível avaliar o desempenho de um determinado sistema, bem como comparar o desempenho de dois ou mais sistemas computacionais. Geralmente, as métricas de desempenho estão relacionadas com requisitos de velocidade, precisão, custo e disponibilidade de serviços (JAIN, 1991).

A escolha adequada das métricas de desempenho está diretamente relacionada com o tipo de sistema que será analisado, ou seja, as medidas de desempenho mais interessantes e os requerimentos operacionais do sistema computacional dependem essencialmente do domínio da aplicação (KENT, 1993). Uma forma de selecionar corretamente as métricas de desempenho, sugerida por Jain (1991), é relacioná-las com os serviços oferecidos pelo sistema.

Nesta dissertação foram utilizadas para mensurar a efetividade do compartilhamento de banda em uma rede SDN as seguintes métricas: (1) Número de pacotes transmitidos para os casos de com e sem barganha; (2) Número de pacotes descartados e; (3) *Jitter*. Estas métricas também são apresentadas na Tabela 8 e foram usadas em todos os experimentos.

Tabela 8 – Métricas de Desempenho

MÉTRICAS	**DETALHES**
I	Quantidade de pacotes que foram transmitidos com e sem barganha.
II	Quantidade de pacotes que foram descartados com e sem barganha, independentemente do motivo (filas saturadas, falhas de rede ou quaisquer outros meios que possibilitem a perda de pacotes).
III	Variação de latência (*jitter*) nos dados transmitidos com e sem barganha.

Algumas dessas métricas apresentadas estão especificadas no documento técnico *OpenFlow Switch Specification* (OPENFLOW..., 2013). Sua estrutura pode ser vista na Figura 24.

Figura 24 – Estrutura da *queue*, segundo a ONF (2013).

```
struct ofp_queue_stats {
    uint32_t port_no;
    uint32_t queue_id;       /* Queue i.d */
    uint64_t tx_bytes;       /* Number of transmitted bytes. */
    uint64_t tx_packets;     /* Number of transmitted packets. */
    uint64_t tx_errors;      /* Number of packets dropped due to overrun. */
    uint32_t duration_sec;   /* Time queue has been alive in seconds. */
    uint32_t duration_nsec;  /* Time queue has been alive in nanoseconds beyond
                                duration_sec. */
};
```

5.2.2 Definição da Carga de Trabalho

De forma sucinta, para geração da carga de trabalho foi usada a ferramenta de geração de tráfego Iperf[1] (IPERF, 2017), configurada no modo *Transport Control Protocol* (TCP) na porta 5002 para produção do fluxo a ser transmitido pelo primeiro *slice* e no modo *User Datagram Protocol* (UDP) na porta 5001 para o fluxo do segundo, como pode ser visto nas Figuras 25 e 26. É importante salientar que tais configurações são específicas para o experimento de validação.

Figura 25 – Configuração do Servidor TCP no Iperf - Experimento de Validação.

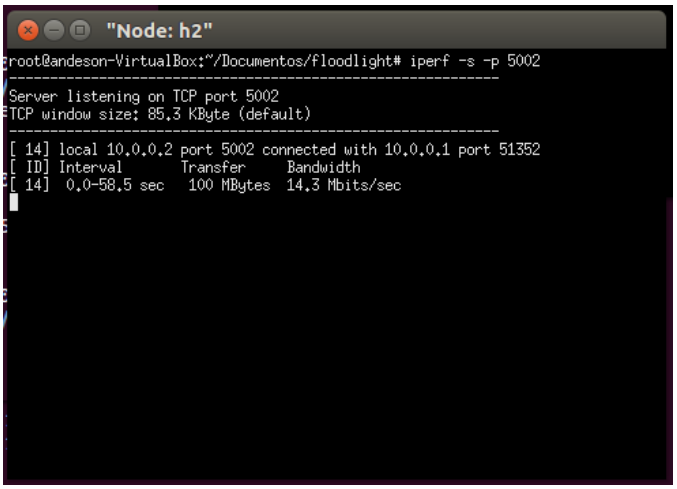
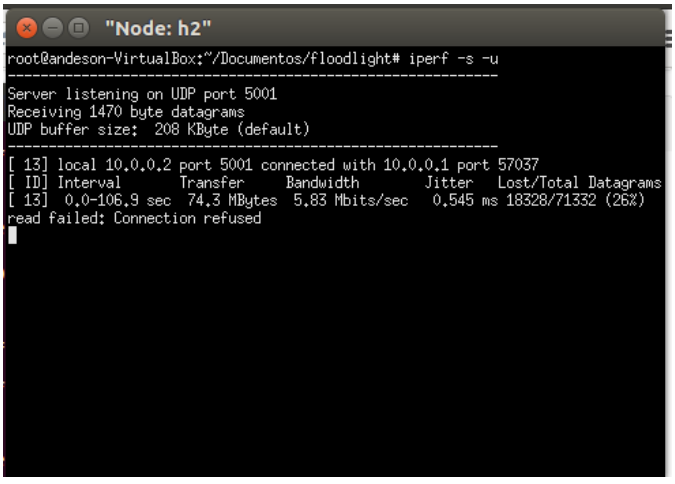


Figura 26 – Configuração do Servidor UDP no Iperf - Experimento de Validação.



Com base no trabalho de GOMES et al. (2013), foi gerada uma carga de 100 MB de dados entre os *hosts* h1 e h2 como pode se visto nas Figuras 27 e 28 dos respectivos clientes Iperf.

[1] IPerf é uma ferramenta útil para administradores de rede, estudantes e pesquisadores, do tipo cliente/servidor, desenvolvida com código livre e gratuita.

Figura 27 – Configuração do Cliente TCP no Iperf - Experimento de Validação.

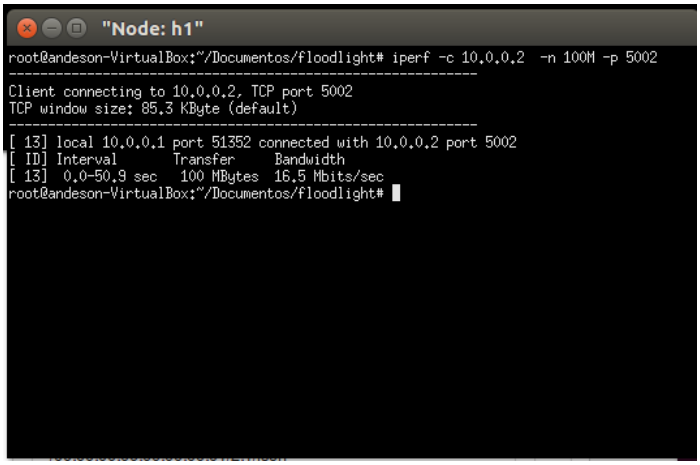
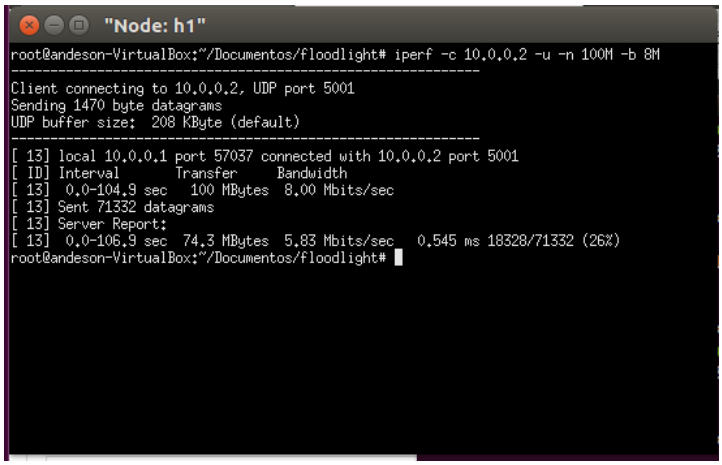


Figura 28 – Configuração do Cliente UDP no Iperf - Experimento de Validação.



Outro importante fator a ser observado na Figura 28 é a definição da largura de banda de 8Mbps, valor este que excede o valor máximo definido na política de QoS criada dentro do OvS como vista na Figura 18 e especificada na Tabela 6, a fim de provocar o compartilhamento de banda.

Já para o experimento do estudo de caso, o Iperf possui as seguintes configurações: o protocolo UDP foi escolhido, uma vez que o mesmo não possui qualquer controle de congestionamento, o que beneficia o uso do mecanismo de compartilhamento de banda. Os fluxos de dados são enviados através das portas 5051, 5052 e 5053 como pode ser visto nas Figuras 29, 30 e 31.

Figura 29 – Configuração do Servidor UDP no Iperf na Porta: 5051 - Experimento do Estudo de Caso.

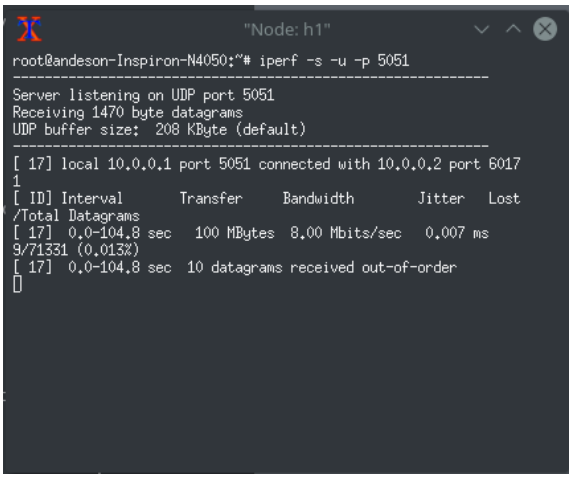


Figura 30 – Configuração do Servidor UDP no Iperf na Porta: 5052 - Experimento do Estudo de Caso.

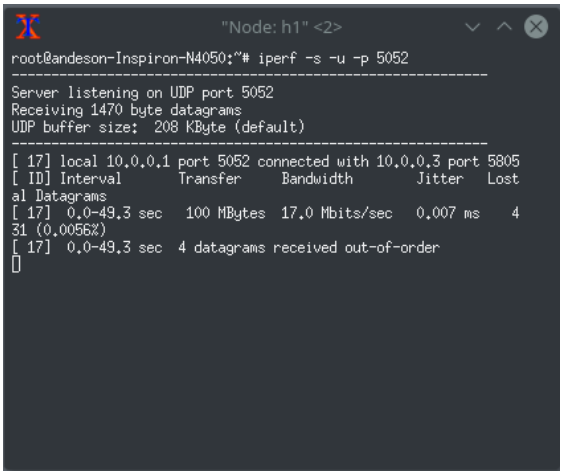
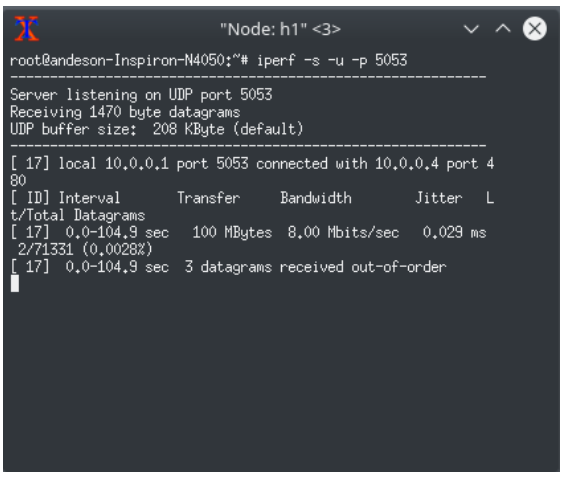


Figura 31 – Configuração do Servidor UDP no Iperf na Porta: 5053 - Experimento do Estudo de Caso.



Ainda com base no trabalho de GOMES et al. (2013), foi gerado o mesmo volume de carga que no experimento de validação e submetido para todos os *hosts* como pode ser visto nas Figuras 32, 33 e 34 dos respectivos clientes Iperf.

Figura 32 – Configuração do Cliente UDP no Iperf na Porta: 5051 - Experimento do Estudo de Caso.

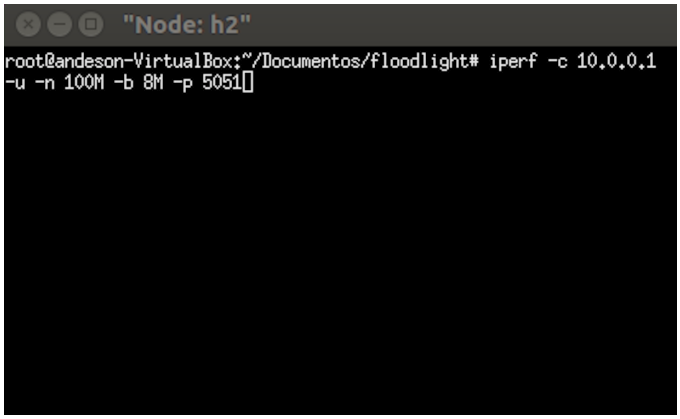


Figura 33 – Configuração do Cliente UDP no Iperf na Porta: 5052 - Experimento do Estudo de Caso.

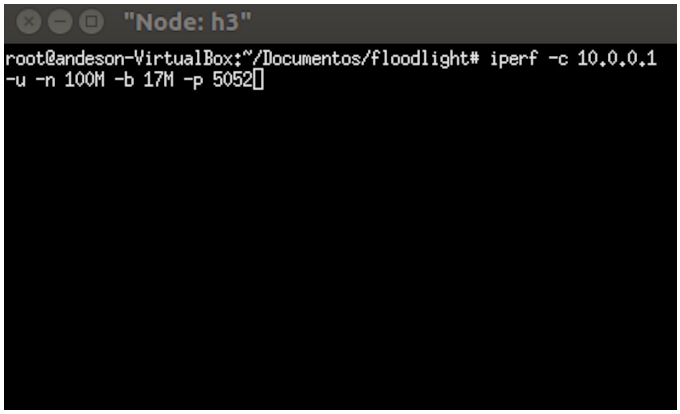
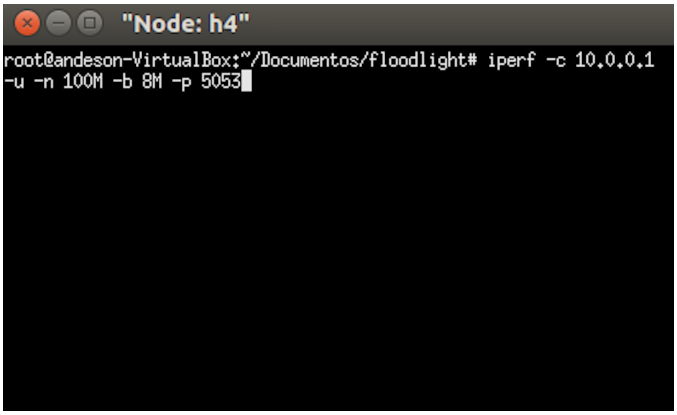


Figura 34 – Configuração do Cliente UDP no Iperf na Porta: 5053 - Experimento do Estudo de Caso.



Outro importante fator a ser observado na Figura 32 é a definição da largura de banda de 8Mbps para a porta 5051, valor este que não excede o valor máximo definido na política de QoS criada dentro do OvS como vista na Figura 21 e especificada na Tabela 7, a fim de não provocar o compartilhamento de banda. Já nas Figuras 33 e 34 os valores excedem o valor máximo definido na política de QoS, o que obriga o compartilhamento de banda.

5.3 Experimento de Validação

Esta seção apresenta os resultados do experimento de validação para os cenários (i) - Sem Barganha e (ii) - Com Barganha, bem como realiza uma análise comparativa entre os resultados dos cenários.

5.3.1 Resultados

Os resultados são apresentados nas formas de gráficos e tabelas para melhor interpretação dos dados expostos.

5.3.1.1 Cenário I - Sem Barganha

Compreende-se como Cenário I a execução do experimento sem qualquer interferência do serviço de compartilhamento de banda. Os gráficos representados nas Figuras 35 e 36, retratam os pacotes enviados e descartados, bem como o seu desempenho.

Figura 35 – Pacotes enviados.

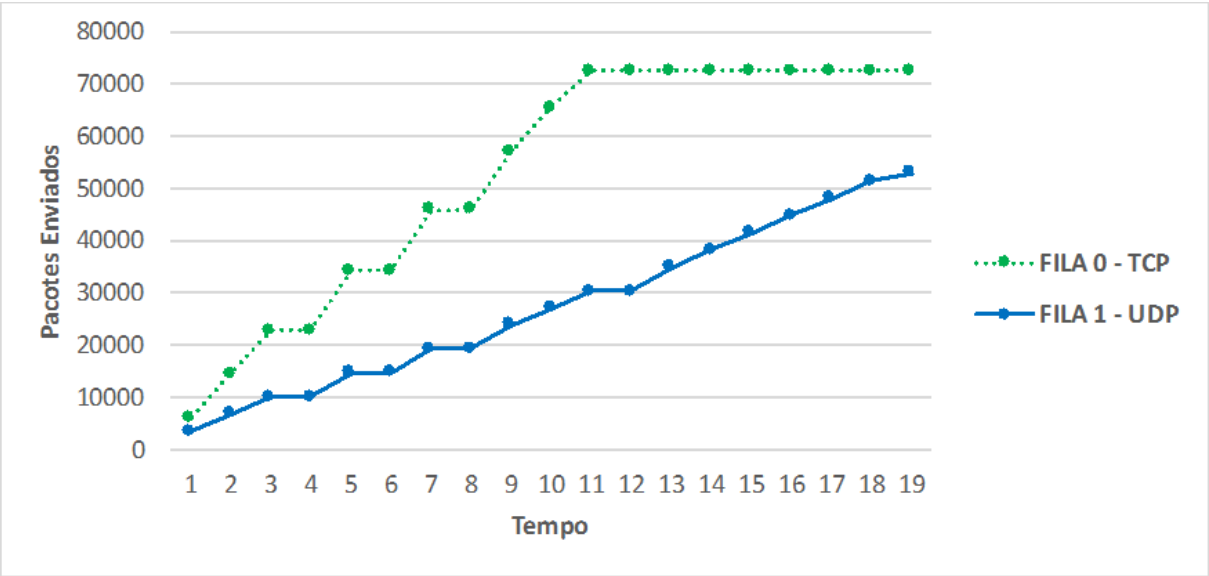
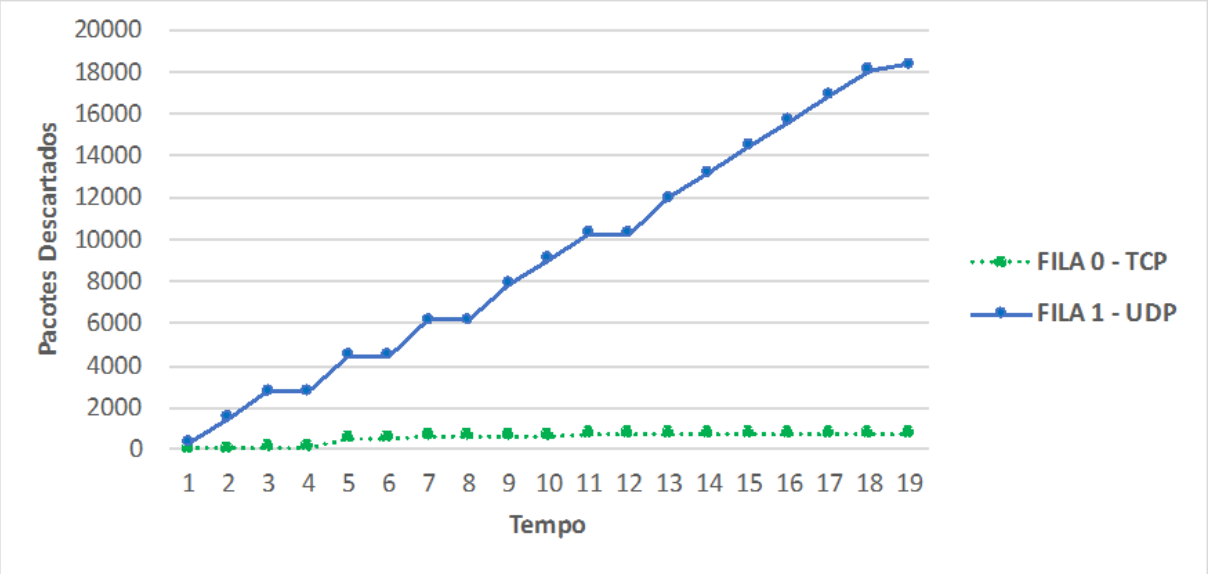


Figura 36 – Pacotes descartados.



Os gráficos representados nas Figuras 35 e 36, apresentam suas curvas pontilhada e linear para cada tipo de protocolo TCP e UDP, respectivamente. Observa-se ao longo da execução do experimento, que o TCP possui o envio de pacotes mais eficiente quando comparado com o UDP. Este, segundo Kurose e Ross (2010), Tanenbaum (2011) e Stallings (2013), não possui controles e mecanismos de fluxos e de congestionamentos, influenciando diretamente na quantidade de perdas de pacotes quando comparado com o TCP. Observa-se ainda o comportamento "serrilhado"e característico do mecanismo de funcionamento do TCP, pois possui tais mecanismos de controles.

De modo análogo, pode ser constatado nos gráficos representados nas Figuras 37 e 38 que o protocolo UDP possui uma perda de pacotes mais acentuada quando comparado com o TCP.

Figura 37 – Percentual de pacotes TCP Enviados e Descartados - Sem Barganha.

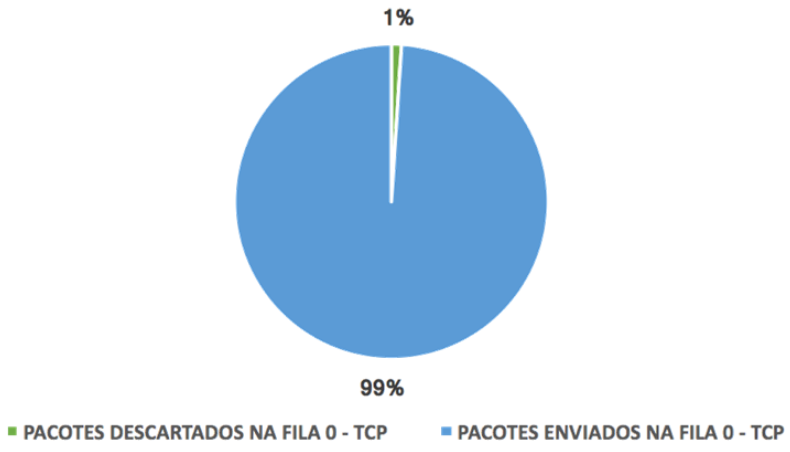
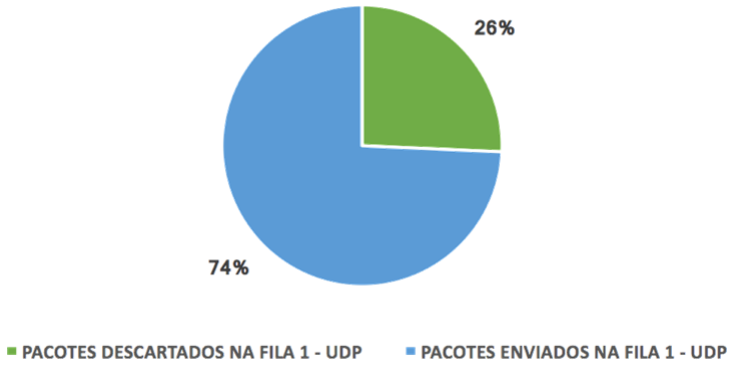


Figura 38 – Percentual de pacotes UDP Enviados e Descartados - Sem Barganha.



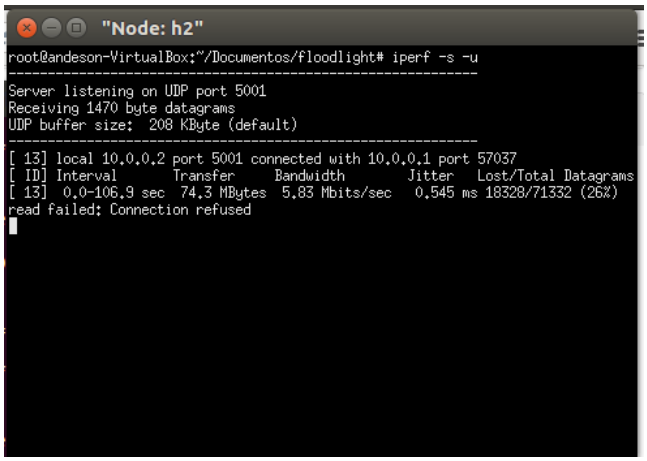
A Tabela 9 apresenta os percentuais.

Tabela 9 – Dados Estatísticos do Cenário I

PROTOCOLO	**DESCRIÇÃO**
TCP	• Pacotes Descartados: 1%; • Pacotes Enviados: 99%.
UDP	• Pacotes Descartados: 26%; • Pacotes Enviados: 74%; • *Jitter*: 0.545 ms%.

A Figura 39, apresenta dentre outras opções, no término da execução do experimento, a variação estatística do atraso (*Jitter*) na entrega de dados.

Figura 39 – Resultado do servidor UDP no Iperf - Sem Barganha.



5.3.1.2 Cenário II - Com Barganha

Compreende-se como Cenário II a execução do experimento utilizando o serviço de compartilhamento de banda. Os gráficos representados nas Figuras 40 e 41, demonstram os pacotes enviados e descartados, bem como o seu desempenho.

Figura 40 – Pacotes enviados.

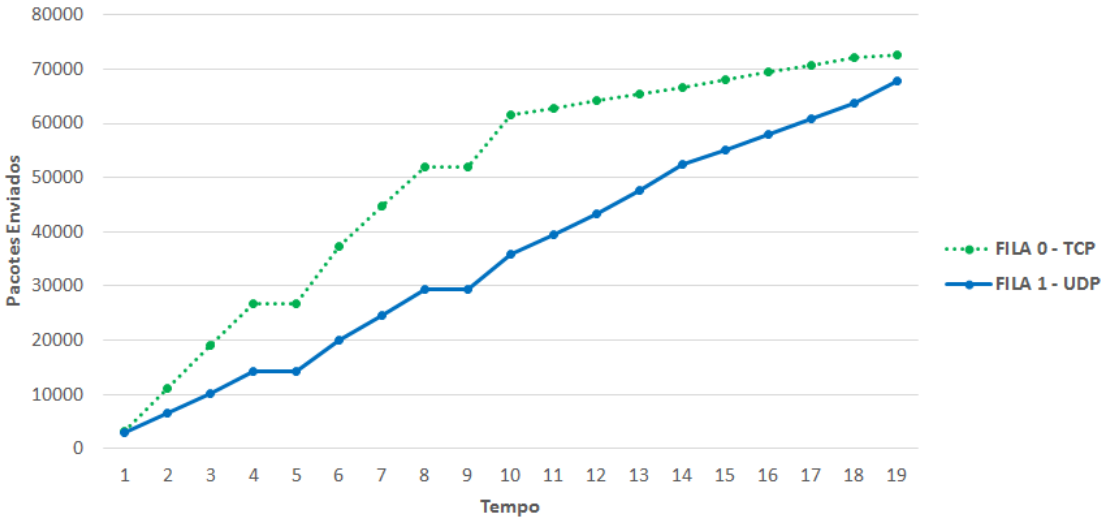
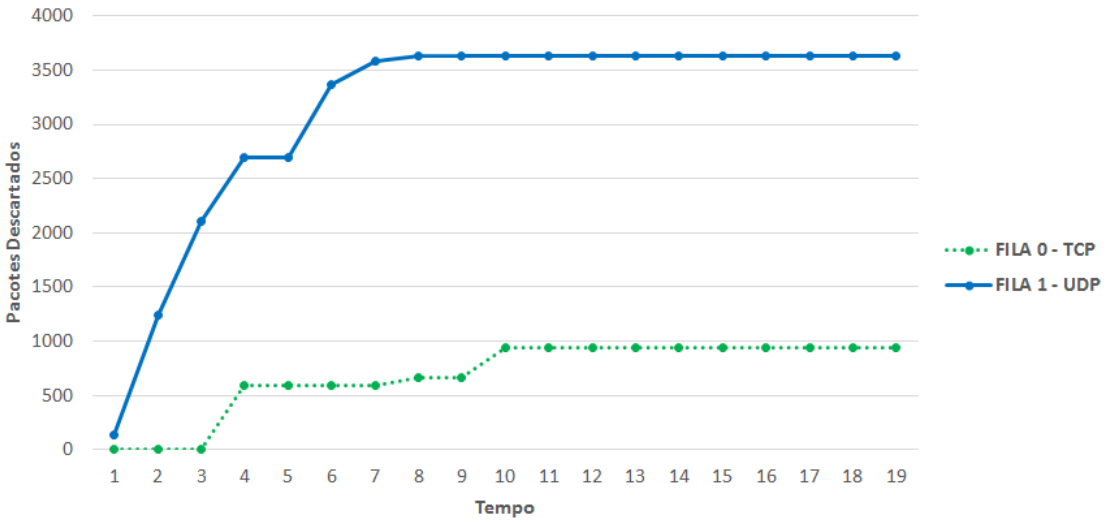


Figura 41 – Pacotes descartados.



Os gráficos representados nas Figuras 40 e 41, apresentam também curvas pontilhada e linear para cada tipo de protocolo TCP e UDP, respectivamente. Observa-se ao longo da execução do experimento, que o UDP possuiu um desempenho bem aproximado do TCP no tocante ao envio de pacotes, pois houve uma redução no número de pacotes descartados. Isso se deu pela adoção do uso do serviço de compartilhamento de banda, que influenciou diretamente em seu desempenho, preservando suas características ora mencionadas anteriormente.

De modo similar, pode ser constatado nos gráficos representados nas Figuras 42 e 43, que o protocolo UDP possui um percentual maior perda de pacotes quando comparado com o TCP. A Tabela 10 apresenta os percentuais.

Figura 42 – Percentual de pacotes TCP Enviados e Descartados - Com Barganha.

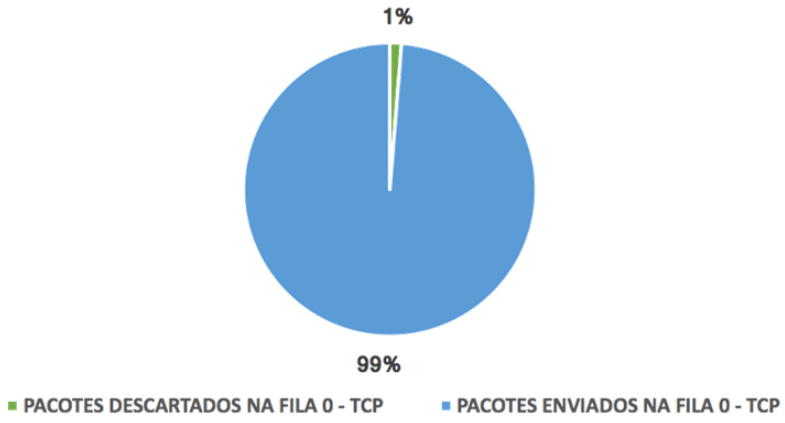


Figura 43 – Percentual de pacotes UDP Enviados e Descartados - Com Barganha.

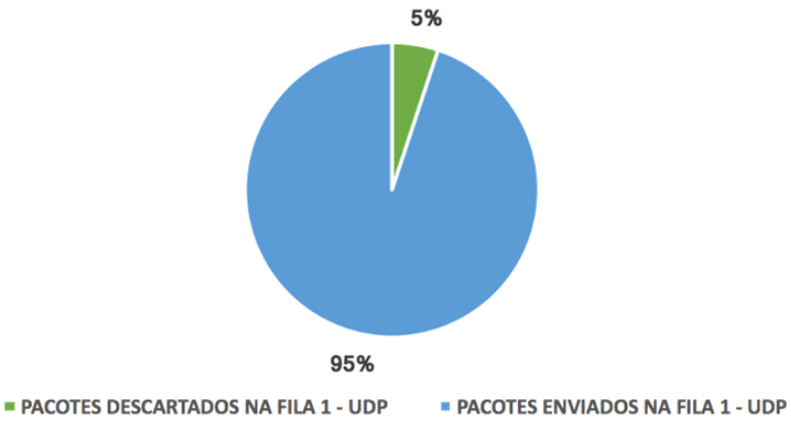
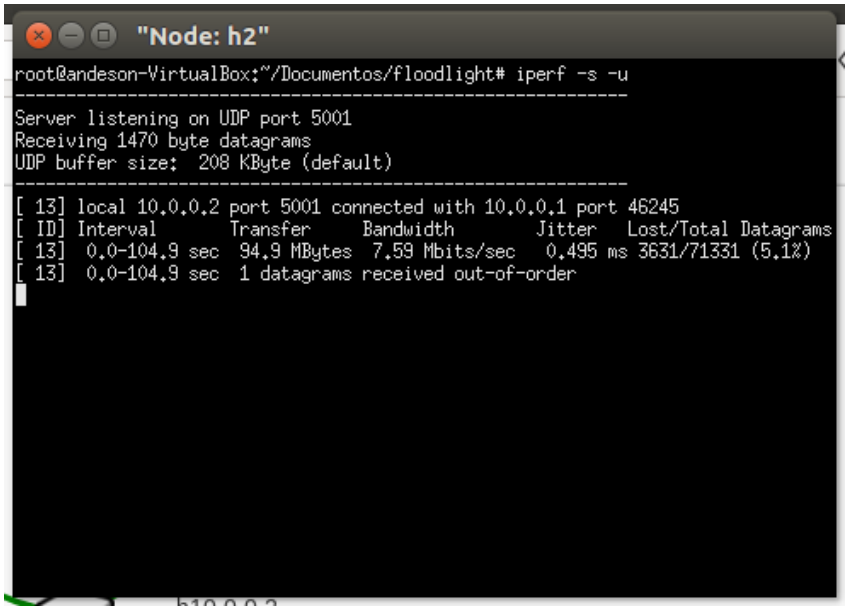


Tabela 10 – Dados Estatísticos do Cenário II

PROTOCOLO	**DESCRIÇÃO**
TCP	• Pacotes Descartados: 1%; • Pacotes Enviados: 99%.
UDP	• Pacotes Descartados: 5%; • Pacotes Enviados: 95%; • *Jitter*: 0.495 ms%.

Ao término do experimento, a Figura 44 apresenta dentre outras opções o valor do (*Jitter*), que é mostrado na Tabela 10.

Figura 44 – Resultado do servidor UDP no Iperf - Com Barganha.



5.3.2 Análise Comparativa dos Resultados

Apresenta-se a análise comparativa dos resultados para os cenários (i) e (ii) definidos no experimento do estudo de caso de validação, ilustrando melhor a relação entre eles, suas particularidade e desempenhos quando comparados.

5.3.2.1 Pacotes enviados

Na Figura 45, apresenta-se uma análise comparativa dos pacotes enviados utilizado nos dois cenários. Observa-se ao longo da execução do experimento, uma equidade na taxa de envio de pacotes para o protocolo TCP (linhas com marcadores [×] e [▲]), contudo, para o UDP (linhas com marcadores [♦] e [•]) é importante ressaltar que o envio de pacotes, utilizando o serviço de compartilhamento de banda, obteve um melhor desempenho quando comparados os métodos. Esta análise comparativa ainda pode ser demonstrada em termos percentuais, a fim de obter uma melhor compreensão como demonstrada no gráfico representado na Figura 46. A Tabela 11 apresenta os percentuais.

Figura 45 – Evolução dos envios de pacotes.

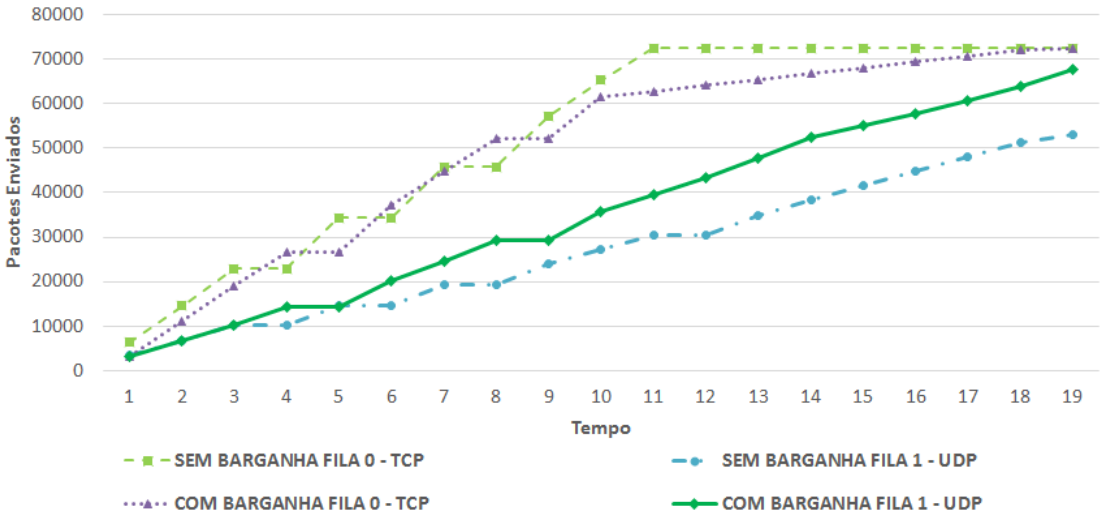


Figura 46 – Percentual geral de pacotes enviados.

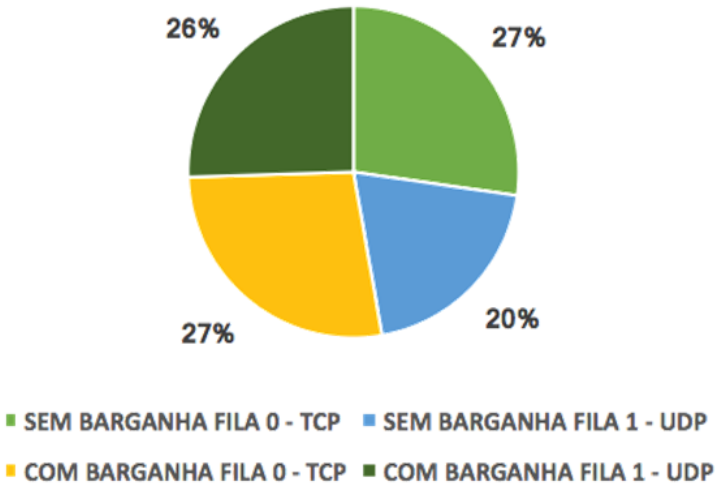


Tabela 11 – Dados estatísticos da análise comparativa dos resultados entre os modelos com e sem compartilhamento de banda - Pacotes Enviados

PROTOCOLO	**DESCRIÇÃO**
TCP	• Pacotes Enviados Sem Barganha: 27%; • Pacotes Enviados Com Barganha: 27%.
UDP	• Pacotes Enviados Sem Barganha: 20%; • Pacotes Enviados Com Barganha: 26%.

5.3.2.2 Pacotes Descartados

No tocante a perda de pacotes, a análise dos resultados demonstram ser ainda mais eficiente quando o serviço de compartilhamento de banda é adotado. O gráfico representado na Figura 47, apresenta no eixo das abscissas a relação crescente de tempo às perdas de pacotes, onde fica evidente que o protocolo UDP sem barganha, representado pelo marcador [•] possui um descarte de pacotes 5 vezes superior quando comparado com a utilização do serviço de compartilhamento de banda. A fim de obter uma melhor visualização desta análise, o gráfico representado na Figura 48, apresenta a comparação dos resultados em percentuais, onde pode-se observar o percentual de 78% dos pacotes são descartados quando o compartilhamento de banda não é usado. A Tabela 12 apresenta os percentuais de pacotes descartados.

Figura 47 – Evolução dos descartes de pacotes.

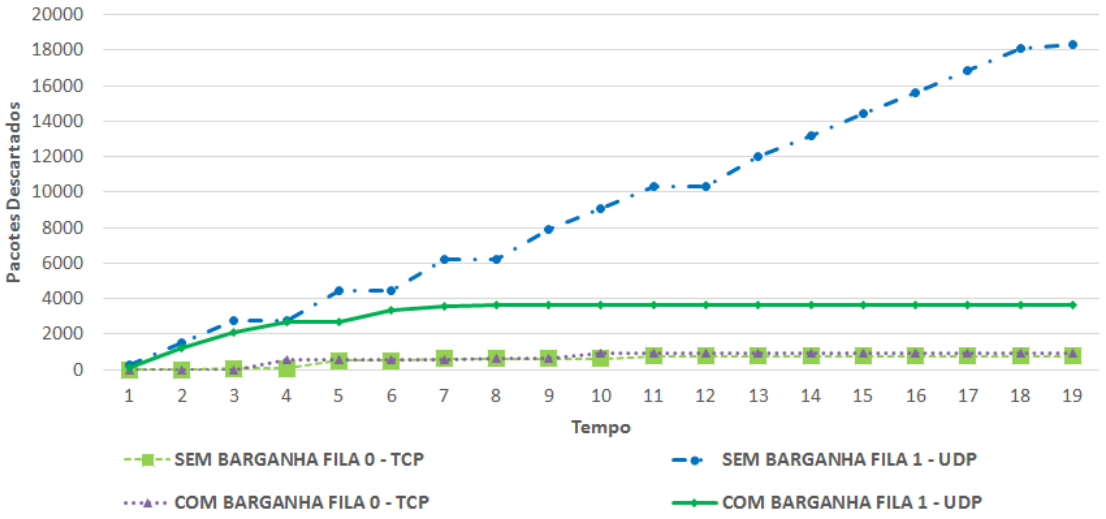


Figura 48 – Percentual geral de pacotes descartados.

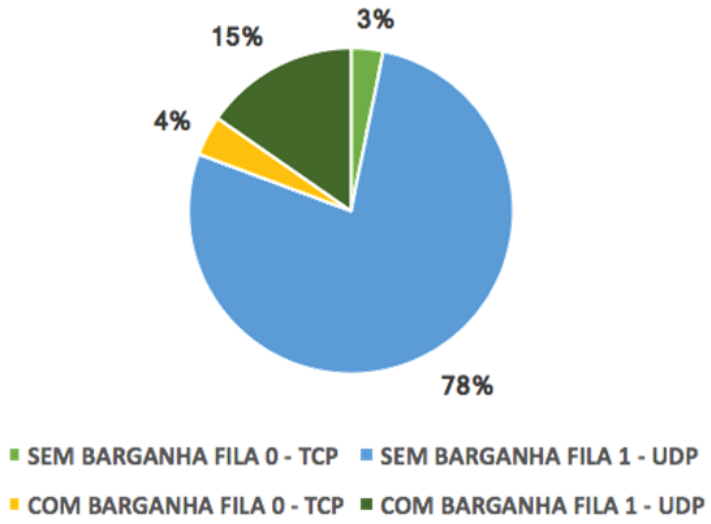


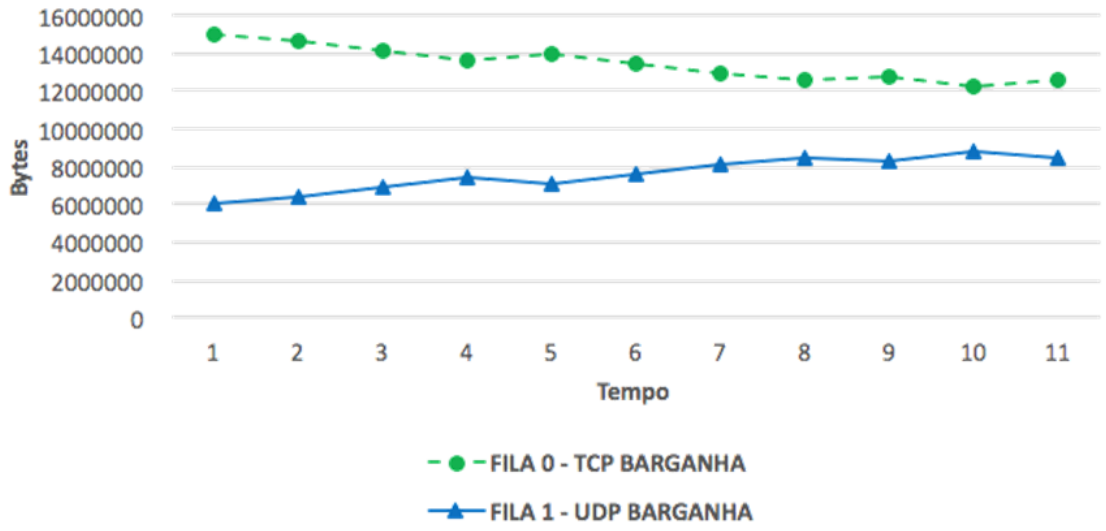
Tabela 12 – Dados estatísticos da análise comprativa dos resultados entre os modelos com e sem compartilhamento de banda - Pacotes Descartados

PROTOCOLO	DESCRIÇÃO
TCP	• Pacotes Descartados Sem Barganha: 3%; • Pacotes Descartados Com Barganha: 4%.
UDP	• Pacotes Descartados Sem Barganha: 78%; • Pacotes Descartados Com Barganha: 15%.

5.3.2.3 Solicitações de Empréstimos

Na Figura 49 apresenta-se todos os momentos em que houve efetivamente solicitações de empréstimos por parte dos protocolos TCP e UDP. Observa-se no gráfico que o protocolo que demandou mais pedidos foi o UDP. O TCP solicitou empréstimos nos momentos 5 e 9 como demonstrados no gráfico.

Figura 49 – Pedido de Empréstimo.



5.4 Experimento do Estudo de Caso

Esta seção apresenta os resultados do experimento do estudo de caso para os cenários (i) - Sem Barganha e (ii) - Com Barganha, bem como realiza uma análise comparativa entre os resultados dos cenários.

5.4.1 Resultados

Os resultados também são apresentados nas formas de gráficos e tabelas para melhor interpretação dos dados expostos.

5.4.1.1 Cenário I - Sem Barganha

Compreende-se como Cenário I a execução do experimento sem qualquer interferência do serviço de compartilhamento de banda. Os gráficos representados nas Figuras 50 e 51, retratam os pacotes enviados e descartados, bem como o seu desempenho.

Figura 50 – Pacotes enviados - Sem Barganha.

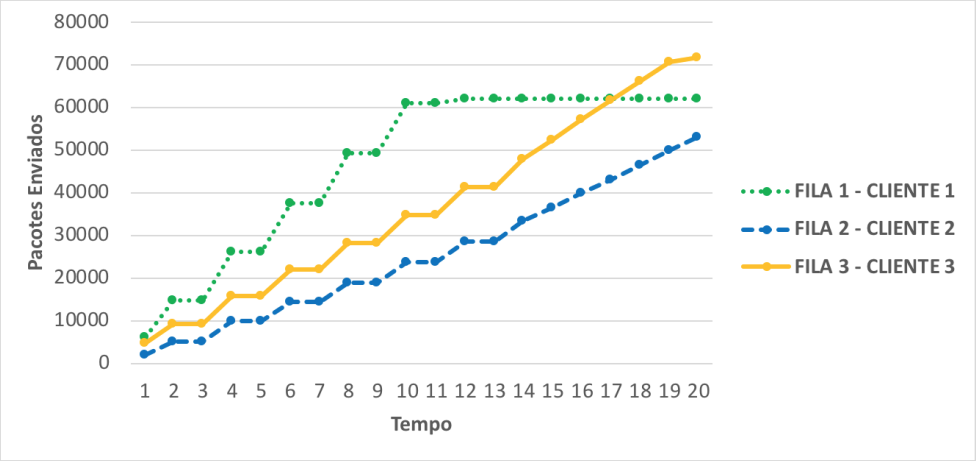
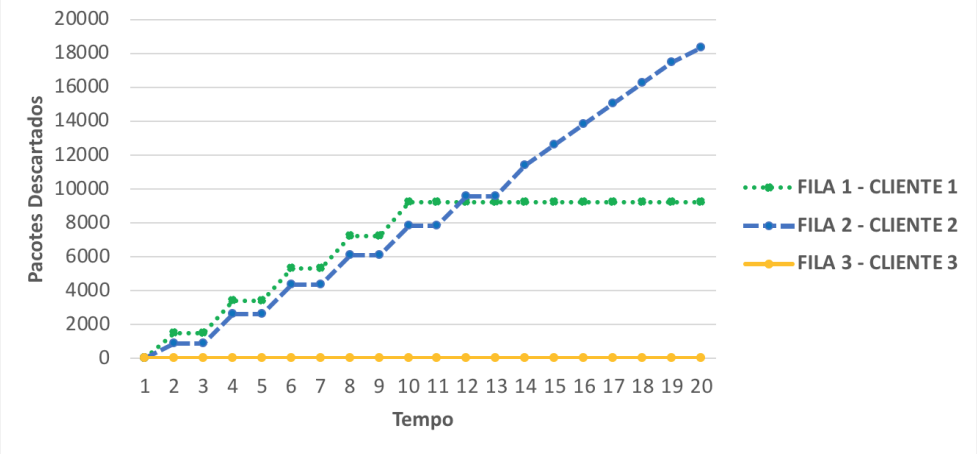


Figura 51 – Pacotes descartados - Sem Barganha.



Os gráficos representados nas Figuras 50 e 51, apresentam suas curvas pontilhada para o cliente 1, tracejada para cliente 2 e linear para o cliente 3. Observa-se na Figura 50 o desempenho dos pacotes enviados por cada cliente. O cliente que obteve o melhor desempenho foi o de número 3, em seguida o pelo de número 1 e por fim o número 2. Já na Figura 51 o desempenho dos pacotes descartados é apresentado também para cada cliente. O cliente que obteve o maior número de descartes foi o cliente 2, em seguida o cliente 1 e por fim o cliente 3.

De modo análogo, pode ser constatado nos gráficos representados nas Figuras 52, 53 e 54 que o cliente 3 obteve o melhor desempenho geral, seguindo pelo cliente 1 e depois o 3.

Figura 52 – Percentual de Pacotes Enviados e Descartados pelo Cliente 1 - Sem Barganha.

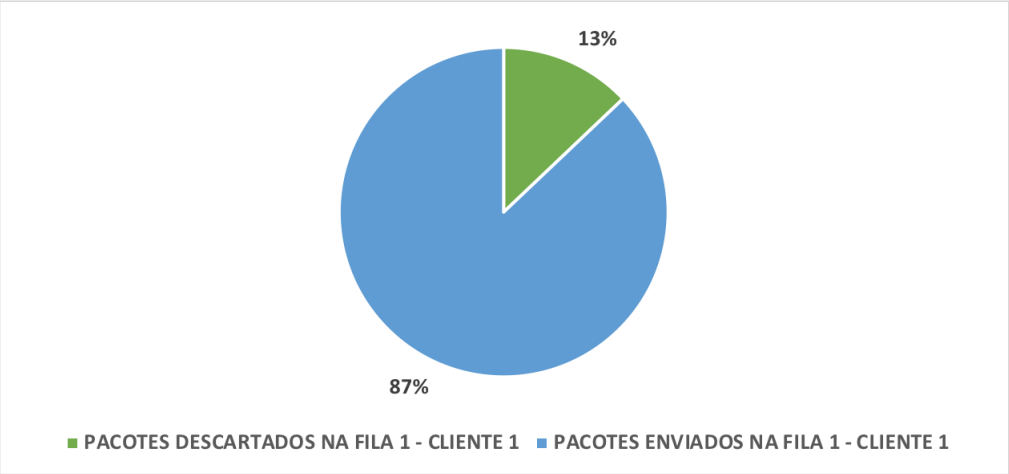


Figura 53 – Percentual de Pacotes Enviados e Descartados pelo Cliente 2 - Sem Barganha.

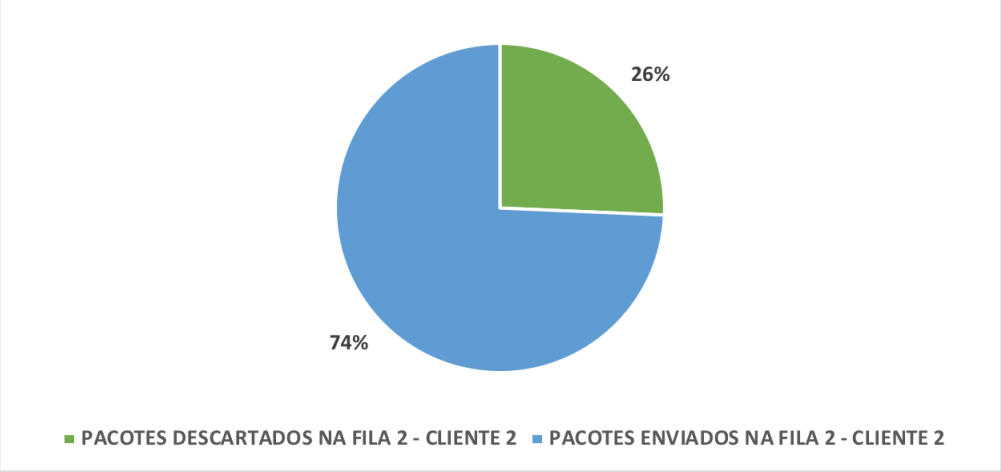
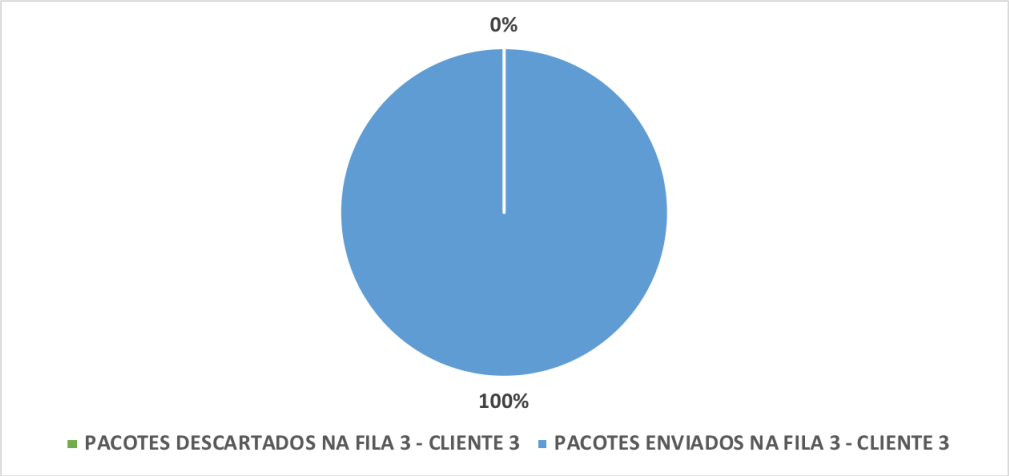


Figura 54 – Percentual de Pacotes Enviados e Descartados pelo Cliente 3 - Sem Barganha.



A Tabela 13 apresenta os percentuais.

Tabela 13 – Dados Estatísticos do Cenário I do Estudo de Caso

CLIENTE	**DESCRIÇÃO**
1	• Pacotes Descartados: 13%; • Pacotes Enviados: 87%; • *Jitter*: 0.007 ms%.
2	• Pacotes Descartados: 26%; • Pacotes Enviados: 74%; • *Jitter*: 0.007 ms%.
3	• Pacotes Descartados: 0%; • Pacotes Enviados: 100%; • *Jitter*: 0.029 ms%.

As Figuras 29, 30 e 31 apresentam dentre outras opções, no término da execução do experimento, a variação estatística do atraso (*Jitter*) na entrega de dados.

5.4.1.2 Cenário II - Com Barganha

Compreende-se como Cenário II a execução do experimento utilizando o serviço de compartilhamento de banda. Os gráficos representados nas Figuras 55 e 56, demonstram os pacotes enviados e descartados, bem como o seu desempenho. É importante ressaltar que, o cliente 3 independentemente do cenário não compartilha sua banda, mesmo que ainda fique com a banda totalmente ociosa, pois trata-se de um cliente prioritário.

Figura 55 – Pacotes enviados - Com Barganha.

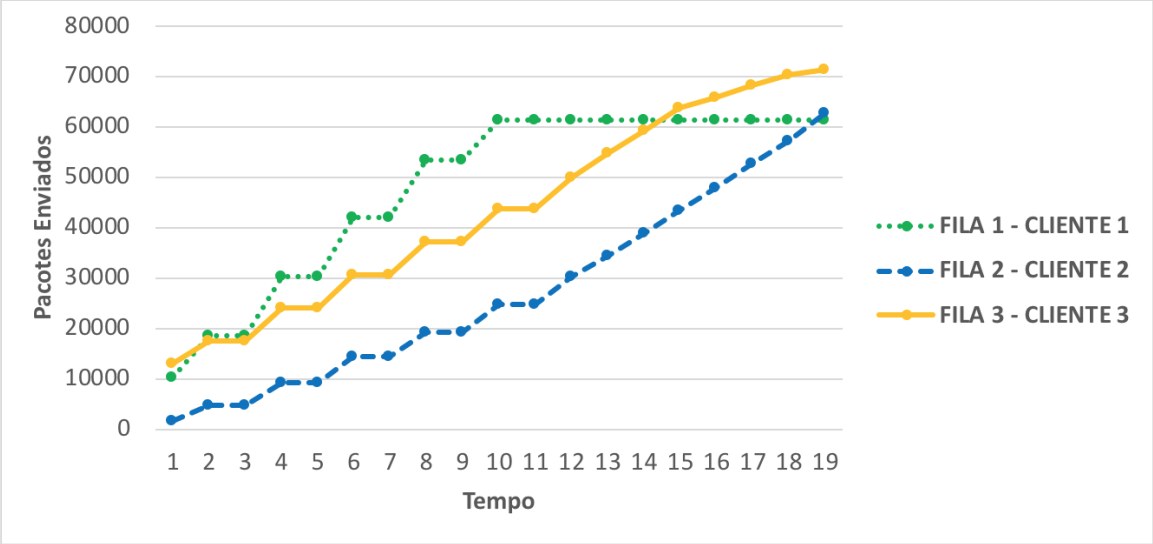
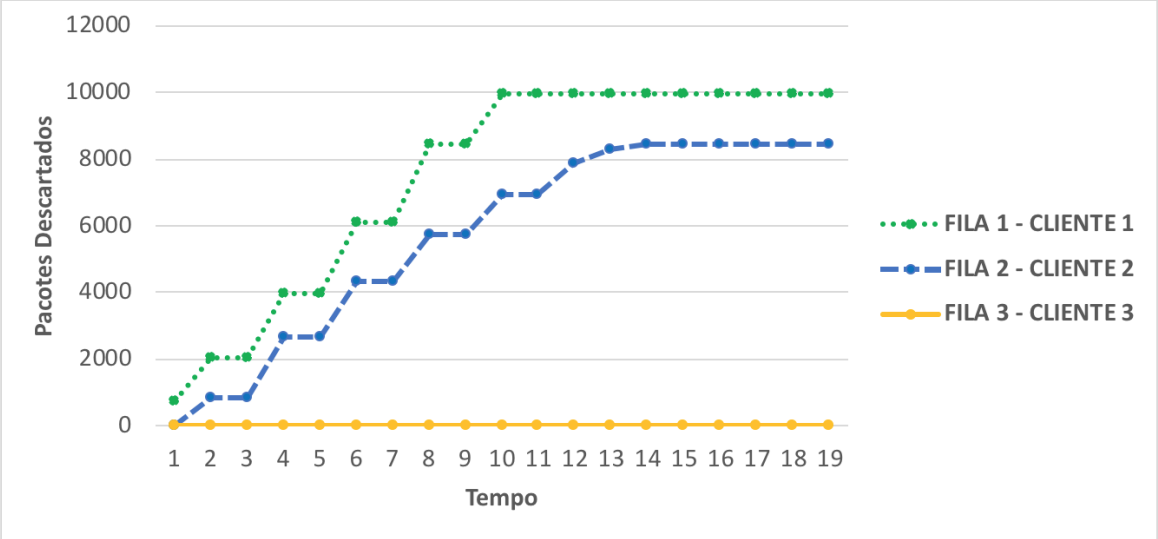


Figura 56 – Pacotes descartados - Com Barganha.



Os gráficos representados nas Figuras 55 e 56, apresentam suas curvas pontilhada para o cliente 1, tracejada para cliente 2 e linear para o cliente 3. Observa-se ao longo da execução do experimento, que o cliente 3 obteve o melhor desempenho no tocante ao número de pacotes enviados, uma vez que não compartilhou sua banda com os demais clientes, pois trata-se de um

cliente com prioridade sobre os demais. Ainda é possível concluir que o mesmo não descartou qualquer pacote, garantindo que toda a demanda de pacotes enviados foram 100% entregues. Contudo, os clientes 1 e 2 obtiveram praticamente o mesmo resultado no final da execução para os pacotes enviados, já para os descartes de pacotes, o cliente que mais descartou foi o cliente 1. Isso se deu pelo fato de possuir uma banda menor que o cliente 2, o que influenciou diretamente em seu desempenho. Tais configurações podem ser vistas na tabela 7.

De modo similar, pode ser constatado nos gráficos representados nas Figuras 57, 58 e 59, que o cliente 1 possuiu o pior desempenho entre os demais. A Tabela 14 apresenta os percentuais.

Figura 57 – Percentual de Pacotes Enviados e Descartados pelo Cliente 1 - Com Barganha.

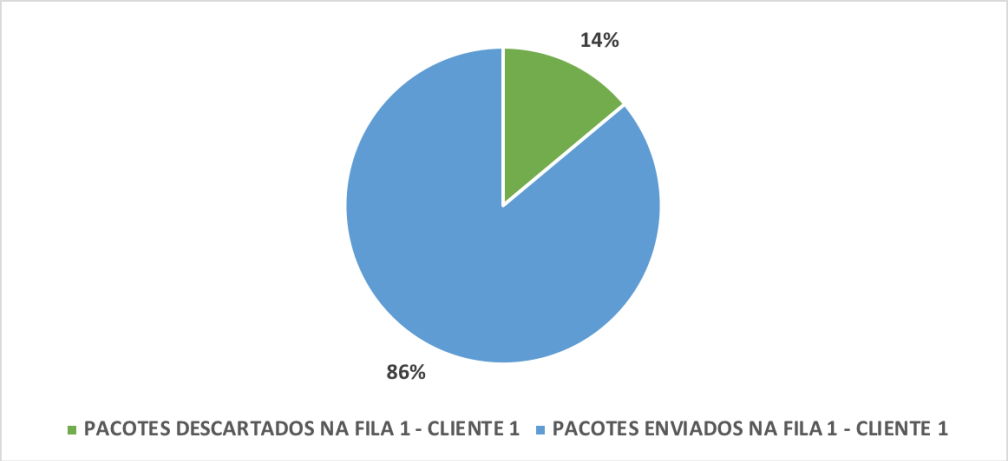


Figura 58 – Percentual de Pacotes Enviados e Descartados pelo Cliente 2 - Com Barganha.

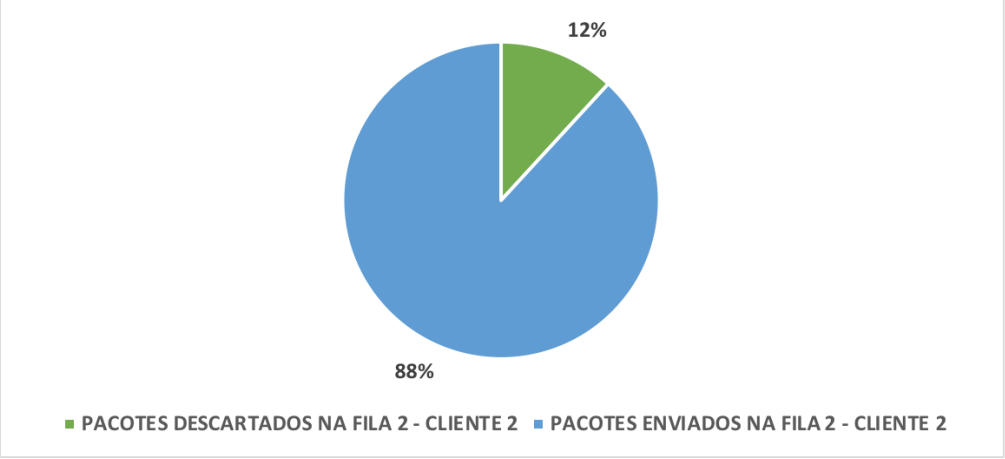


Figura 59 – Percentual de Pacotes Enviados e Descartados pelo Cliente 3 - Com Barganha.

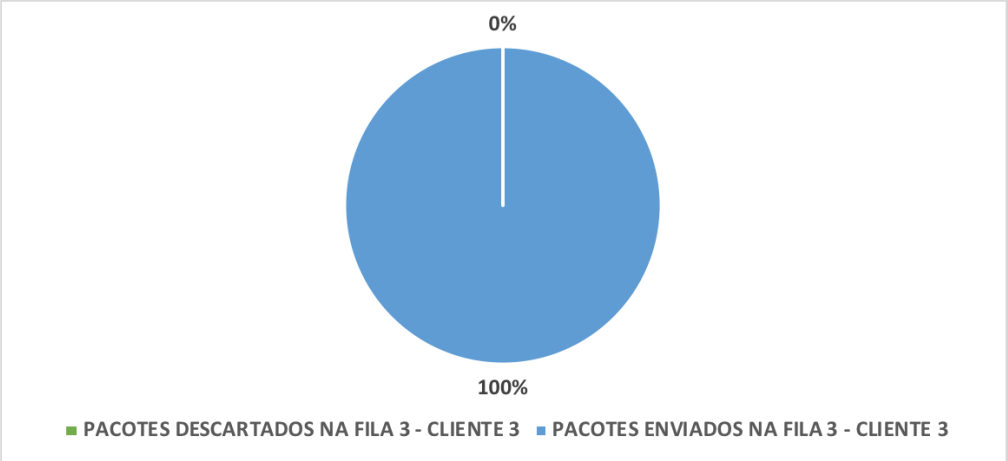


Tabela 14 – Dados Estatísticos do Cenário II do Estudo de Caso

CLIENTE	DESCRIÇÃO
1	• Pacotes Descartados: 14%; • Pacotes Enviados: 86%; • *Jitter*: 13.234 ms%.
2	• Pacotes Descartados: 12%; • Pacotes Enviados: 88%; • *Jitter*: 15.911 ms%.
3	• Pacotes Descartados: 0%; • Pacotes Enviados: 100%; • *Jitter*: 0.016 ms%.

Ao término do experimento, as Figuras 29, 30 e 31 apresentam dentre outras opções o valor do (*Jitter*), que é mostrado na Tabela 14.

5.4.2 Análise Comparativa dos Resultados

Apresenta-se a análise comparativa dos resultados para os cenários (i) e (ii) definidos no experimento do estudo de caso, ilustrando melhor a relação entre eles, suas particularidade e desempenhos quando comparados.

5.4.2.1 Pacotes enviados

Na Figura 60, apresenta-se uma análise comparativa dos pacotes enviados utilizado nos dois cenários. Observa-se ao longo da execução do experimento, uma equidade na taxa de envio de pacotes para o cliente 3, pois o mesmo não faz compartilhamento de banda. Contudo, para os clientes 1 e 2, ficou evidente um leve aumento de desempenho para o cliente 2 quando adotado o serviço de compartilhamento de banda. Esta análise comparativa ainda pode ser demonstrada em termos percentuais, a fim de obter uma melhor compreensão como demonstrada no gráfico representado na Figura 61. A Tabela 15 apresenta os percentuais.

Figura 60 – Evolução dos envios de pacotes.

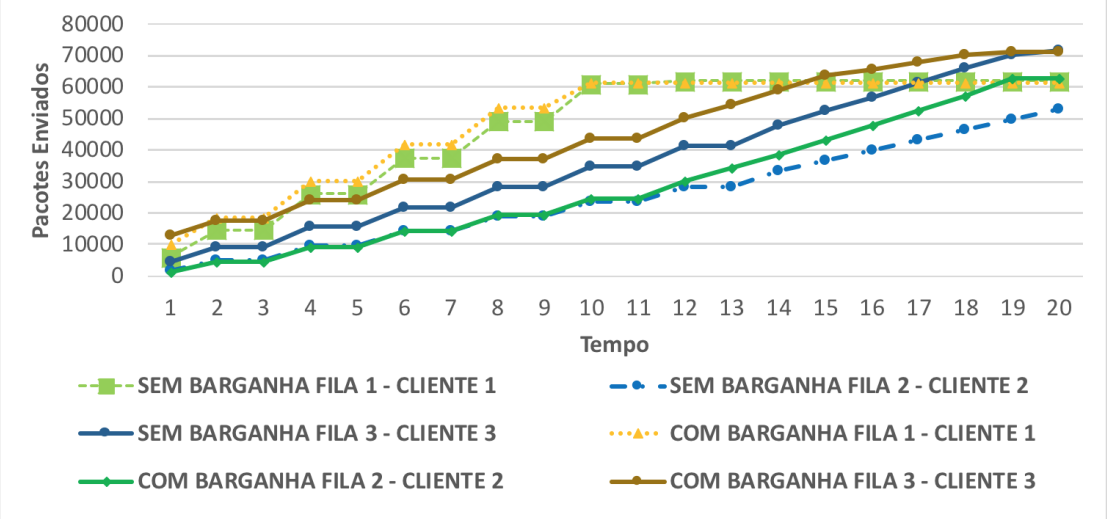


Figura 61 – Percentual geral de pacotes enviados.

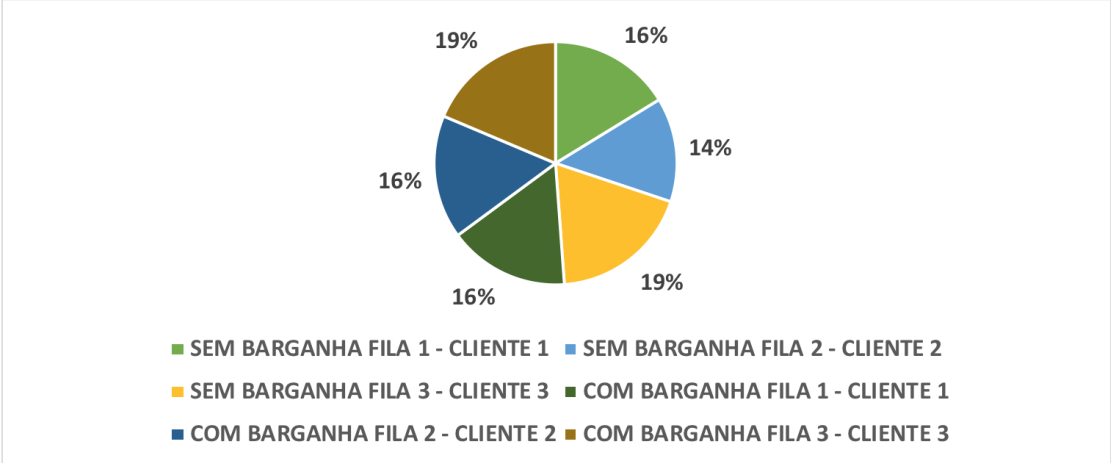


Tabela 15 – Dados estatísticos da análise comparativa dos resultados entre os modelos com e sem compartilhamento de banda do Estudo de Caso- Pacotes Enviados

CENÁRIOS	DESCRIÇÃO
1	• Pacotes Enviados Sem Barganha: 16%; • Pacotes Enviados Com Barganha: 16%.
2	• Pacotes Enviados Sem Barganha: 14%; • Pacotes Enviados Com Barganha: 16%.
3	• Pacotes Enviados Sem Barganha: 19%; • Pacotes Enviados Com Barganha: 19%.

5.4.2.2 Pacotes Descartados

No tocante a perda de pacotes, a análise dos resultados demonstram ser ainda mais eficiente quando o serviço de compartilhamento de banda é adotado. O gráfico representado na Figura 62, apresenta no eixo das abscissas a relação crescente de tempo às perdas de pacotes, onde fica evidente que o cliente 2 sem barganha, possui o dobro de descarte de pacotes quando comparado com a utilização do serviço de compartilhamento de banda. A fim de obter uma melhor visualização desta análise, o gráfico representado na Figura 63, apresenta a comparação dos resultados em percentuais, onde pode-se observar o percentual de 40% dos pacotes são descartados quando o compartilhamento de banda não é usado. A Tabela 16 apresenta os percentuais de pacotes descartados.

Figura 62 – Evolução dos descartes de pacotes.

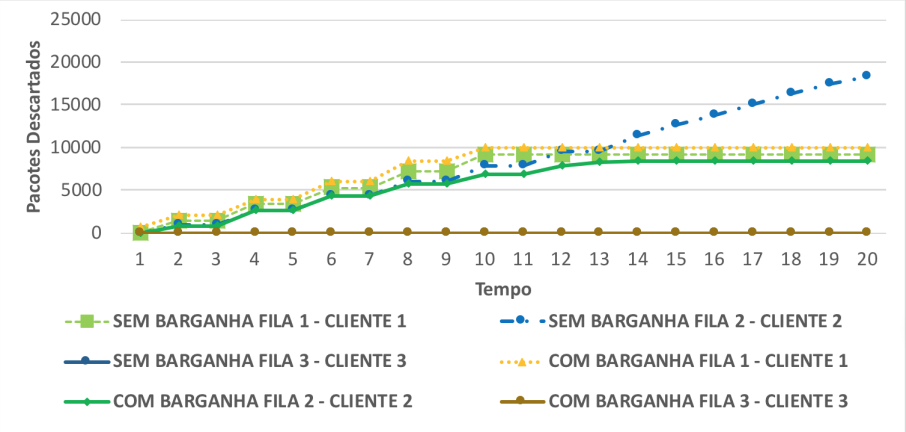


Figura 63 – Percentual geral de pacotes descartados.

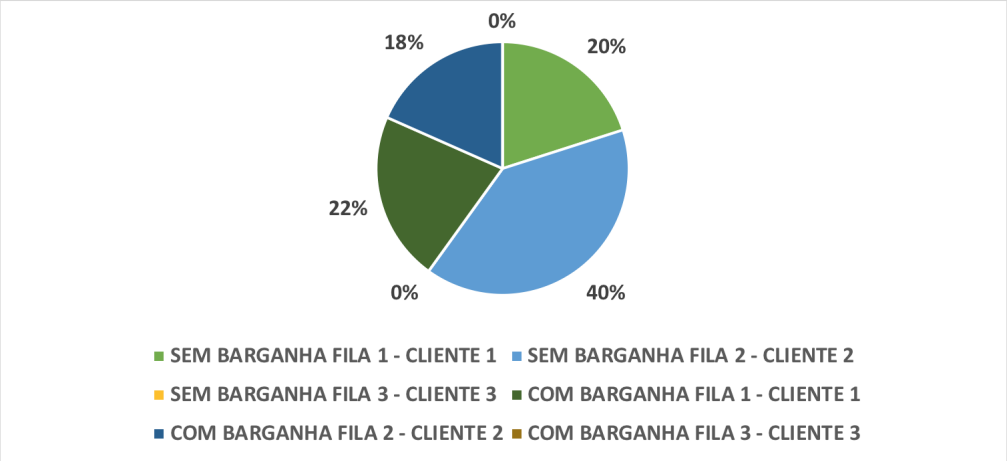


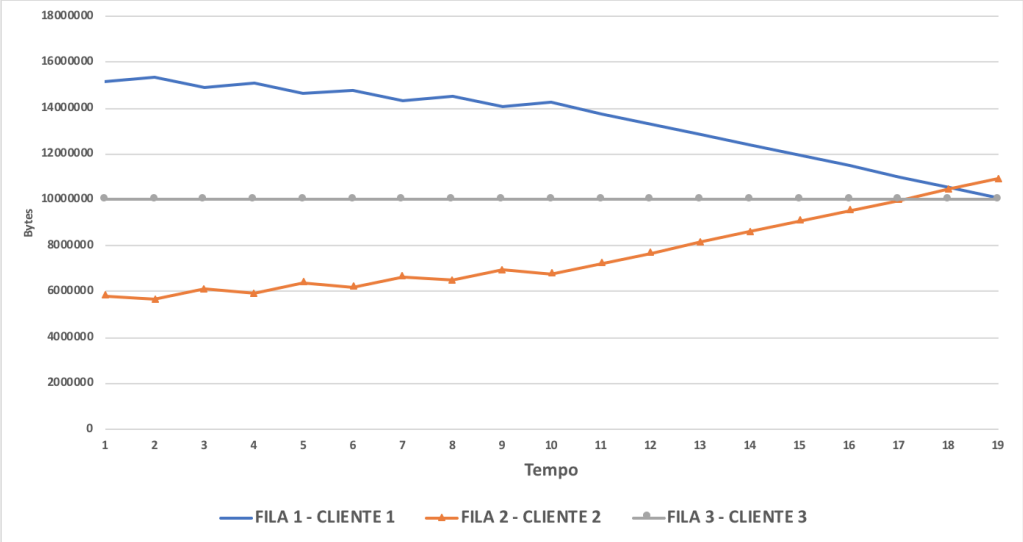
Tabela 16 – Dados estatísticos da análise comprativa dos resultados entre os modelos com e sem compartilhamento de banda - Pacotes Descartados

CENÁRIOS	**DESCRIÇÃO**
1	• Pacotes Descartados Sem Barganha: 20%; • Pacotes Descartados Com Barganha: 22%.
2	• Pacotes Descartados Sem Barganha: 40%; • Pacotes Descartados Com Barganha: 18%.
3	• Pacotes Descartados Sem Barganha: 0%; • Pacotes Descartados Com Barganha: 0%.

5.4.2.3 Solicitações de Empréstimos

Na Figura 64 apresenta-se todos os momentos em que houve efetivamente solicitações de empréstimos por parte dos clientes 1 e 2, uma vez que o cliente 3 não faz uso do mecanismo de compartilhamento de banda, preservando inalterado o tamanho da banda durante todo o estudo de caso. Observa-se ainda que o cliente que demandou mais pedidos de empréstimos foi o cliente 2, como demonstrado no gráfico.

Figura 64 – Pedido de Empréstimo.



6
Conclusão

No presente trabalho foi discutido o compartilhamento de banda em redes definidas por *software* utilizando o protocolo OpenFlow. A partir desse fato, foi proposto uma arquitetura denominada BSS/SDN, projetada em camadas com papeis e responsabilidades definidas que suportasse estruturalmente um serviço de compartilhamento de banda em redes SDN que levasse em consideração diferentes políticas de QoS para classificação de fluxos baseadas ou não em protocolos da camada de transporte. Esse modelo foi implementado, validado e submetido a um estudo de caso em um ambiente de rede virtual, baseado em máquinas virtuais e no Open vSwitch utilizando o protocolo OpenFlow, com o intuito de ser aprimorado e reproduzido no futuro em uma rede operacional real.

Para o ambiente de validação os resultados mostraram que para o protocolo TCP, os resultados não apresentaram diferenças significativas de desempenho quando usado o mecanismo de compartilhamento de banda, uma vez que o mesmo possui mecanismos nativos para controle de fluxo e de congestionamento que promove um efetivo controle da banda consumida em um canal de comunicação. Contudo, com a adoção do compartilhamento de banda para o protocolo UDP os resultados foram bastante satisfatórios, alcançando um percentual de 15% de descarte de pacotes, contra 78% sem a utilização do mesmo, demonstrando um ganho efetivo e real de mais de 5 vezes de uso. Contudo, para o estudo de caso os resultados demonstraram um compartilhamento de efetivo de banda, promovendo um ganho de mais de 100% de pacotes descartados pelo cliente 2 quando adotado o serviço de compartilhamento de banda.

Concluiu-se que o serviço de compartilhamento de banda em redes SDN, usando o protocolo OpenFlow, atingiu um nível de ganho significativo e pode ser um serviço aliado para melhorar a efetividade das políticas de QoS, evitando a saturação das filas.

6.1 Contribuições

As principais contribuições deste trabalho foi: a definição de uma arquitetura de serviço denominada de BSS/SDN (*Bandwidth Sharing Service on Software-Defined Networks*), especificada em camadas que tem como objetivo separar os serviços a serem oferecidos no compartilhamento de banda em redes definidas por *software*, utilizando modelos de barganha.

Outras contribuições deste trabalho foram, desenvolver *software* que aplicasse os conceitos de SDN em um ambiente não simulado, demonstrando sua aplicabilidade em ambientes reais.

6.2 Trabalhos futuros

Como trabalhos futuros sugere-se inicialmente realizar modificações nos parâmetros, como exemplo taxa de barganha dos empréstimos de banda, alterações na topologia e nos cenários utilizados nos experimentos desse trabalho.

Sugere-se também como contribuição para melhoria do desempenho de mecanismo de compartilhamento de banda em redes SDN, a implementação do algoritmo de otimização por colônia de formigas, com o objetivo de melhorar o tempo de reconfiguração dos elementos de rede.

Como trabalhos futuros, aponta-se ainda a possibilidade de mesclar diferentes mecanismos de equidade, adicionando a teoria dos jogos para maximizar a eficiência na gestão das filas.

Desenvolver modelos diferenciados de barganha, a fim de realizar comparações de desempenho entre os modelos.

Referências

A. B. MacKenzie, S. B. W. Game theory in communications: Motivation, explanation, and application to power control. *IEEE GLOBECOM*, v. 2, p. 821–826, 2001. Citado na página 41.

ABRAHAO, D. C. Dissertação de Mestrado, *Alocacao de blocos de recurso em redes LTE e utilizando logica fuzzy e estimacao adaptativa de banda efetiva.* 2015. 129 p. Citado na página 49.

AMIRI, M. et al. An sdn controller for delay and jitter reduction in cloud gaming. In: *Proceedings of the 23rd ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2015. (MM '15), p. 1043–1046. ISBN 978-1-4503-3459-4. Disponível em: <http://doi.acm.org/10.1145/2733373.2806397>. Citado na página 44.

ANTONIOU, J.; PITSILLIDES, A. Supporting Radio Access Network Selection in IMS over 4G: Proposing a Game Theoretic Scheme. *16th IST Mobile and Wireless Communication Summit*, p. 1–6, 2007. Disponível em: <http://www.cs.ucy.ac.cy/ResearchLabs/netrl/papers/files/IEEE_VTM_ON_IMS_jaap.pdf>. Citado na página 41.

AWDUCHE, D. et al. *Requirements for Traffic Engineering Over MPLS*. IETF, 1999. 1–29 p. (Request for Comments, 2702). Disponível em: <http://www.ietf.org/rfc/rfc2702.txt>. Citado na página 20.

BANSAL, M. et al. Openradio: A programmable wireless dataplane. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 109–114. ISBN 978-1-4503-1477-0. Disponível em: <http://doi.acm.org/10.1145/2342441.2342464>. Citado na página 38.

BOSSHART, P. et al. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 3, p. 87–95, jul. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2656877.2656890>. Citado na página 39.

CAESAR, M. et al. Design and implementation of a routing control platform. In: *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*. Berkeley, CA, USA: USENIX Association, 2005. (NSDI'05), p. 15–28. Disponível em: <http://dl.acm.org/citation.cfm?id=1251203.1251205>. Citado na página 27.

CARVALHO, G. Dissertação de Mestrado, *Compartilhamento de Banda em Redes Definidas por Softwares*. 2017. 77 p. Citado na página 47.

CASADO, M. et al. Rethinking enterprise network control. *IEEE/ACM Transactions on Networking*, v. 17, n. 4, p. 1270–1283, Aug 2009. ISSN 1063-6692. Citado na página 27.

CHANDRASEKARAN, B.; BENSON, T. Tolerating sdn application failures with legosdn. In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. New York, NY, USA: ACM, 2014. (HotNets-XIII), p. 22:1–22:7. ISBN 978-1-4503-3256-9. Disponível em: <http://doi.acm.org/10.1145/2670518.2673880>. Citado na página 43.

DORIA, e. a. *General Switch Management Protocol V3*. IETF, 2002. RFC 3292 (Standards Track). (Request for Comments, 3292). Disponível em: <https://www.ietf.org/rfc/rfc3292.txt>. Citado na página 26.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn: An intellectual history of programmable networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 2, p. 87–98, abr. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2602204.2602219>. Citado 3 vezes nas páginas 23, 26 e 27.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. Tr10: Software-defined networking. 2014. Disponível em: <http://www.technologyreview.com/web/22120/>. Citado na página 25.

FERGUSON, A. D. et al. Hierarchical policies for software defined networks. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 37–42. ISBN 978-1-4503-1477-0. Disponível em: <http://doi.acm.org/10.1145/2342441.2342450>. Citado na página 19.

FERNANDES E.L., R. C. S. M. V. A. . V. F. Building upon routeflow: a sdn development experience. In: . [s.n.], 2013. Disponível em: <http://ce-resd.facom.ufms.br/sbrc/2013/artigo-61.pdf>. Citado na página 36.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado), 2000. AAI9980887. Citado na página 50.

G. Piro, L. Grieco, G. Boggia, F. Capozzi, P. C. Simulating lte cellular systems: an open source framework. *IEEE Trans. Veh. Technol.*, v. 60, p. 498–513, 2010. Citado na página 42.

GOMES, V. S. et al. Isolamento de recursos de rede em ambientes virtuais baseados em openflow/sdn. In: . [s.n.], 2013. Disponível em: <http://sbrc2013.unb.br/files/anais/wgrs/artigos/artigo-7.pdf>. Citado 3 vezes nas páginas 58, 67 e 70.

HAMOUDA, S.; Ben Chaabane, I.; TABBANE, S. Cooperative Bandwidth Sharing for Relaying in LTE-Advanced Using Game Theory. *Vehicular Technology, IEEE Transactions on*, v. 64, n. 6, p. 2306–2317, 2015. ISSN 0018-9545. Citado na página 20.

IPERF. *Iperf Iperf*. 2017. Disponível em: <https://iperf.fr/>. Citado na página 67.

ITURRALDE, M. et al. Performance Study of Multimedia Services Using Virtual Token Mechanism for Resource Allocation in LTE Networks. *IEEE Vehicular Technology Conference (VTC Fall)*, p. 1–5, 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6092905>. Citado na página 42.

JAHAN, a. H.; HASSAN, M.; DAS, S. K. A brinkmanship game theory model for competitive wireless networking environment. *IEEE Local Computer Network Conference*, Ieee, p. 120–127, oct 2010. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5735685>. Citado na página 41.

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. [S.l.: s.n.], 1991. 685 p. ISBN 0471503363. Citado na página 66.

JARSCHEL, M. et al. Interfaces, attributes, and use cases: A compass for sdn. *IEEE Communications Magazine*, v. 52, n. 6, p. 210–217, June 2014. ISSN 0163-6804. Citado 2 vezes nas páginas 37 e 39.

JO, E. et al. A simulation and emulation study of sdn-based multipath routing for fat-tree data center networks. In: *Proceedings of the 2014 Winter Simulation Conference*. Piscataway, NJ, USA: IEEE Press, 2014. (WSC '14), p. 3072–3083. Disponível em: <http://dl.acm.org/citation.cfm?id=2693848.2694235>. Citado 2 vezes nas páginas 58 e 59.

JO, J.; LEE, S.; KIM, J. Software-defined home networking devices for multi-home visual sharing. *Consumer Electronics, IEEE Transactions on*, v. 60, n. 3, p. 534–539, Aug 2014. ISSN 0098-3063. Citado na página 19.

KENT, K. Book review: Introduction to computer system performance evaluation by krishna kant (mcgraw-hill, 1992). *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 21, n. 2, p. 7–9, dez. 1993. ISSN 0163-5999. Reviewer-Lipsky, Lester. Disponível em: <http://doi.acm.org/10.1145/174215.1044951>. Citado na página 66.

KHAN, M. a. et al. Cooperation-based resource allocation and call admission for wireless network operators. *Telecommunication Systems*, jan 2011. ISSN 1018-4864. Disponível em: <http://www.springerlink.com/index/10.1007/s11235-010-9412-1>. Citado na página 41.

KHAN, M. A. et al. Cooperative game theoretic approach to integrated bandwidth sharing and allocation. *2009 International Conference on Game Theory for Networks*, Ieee, p. 1–9, may 2009. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5137376>. Citado na página 41.

KHAN, M. A. et al. Network level cooperation for resource allocation in future wireless networks. *2008 1st IFIP Wireless Days*, Ieee, p. 1–5, nov 2008. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4812916>. Citado na página 41.

KIM, K. S. On the Excess Bandwidth Allocation in ISP Traffic Control for Shared Access Networks. *Communications Letters, IEEE*, v. 18, n. 4, p. 692–695, 2014. ISSN 1089-7798. Citado na página 18.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219. Citado 9 vezes nas páginas 8, 23, 25, 28, 34, 36, 37, 38 e 39.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: Uma Abordagem Top-down*. 5ª. ed. [S.l.]: Pearson do Brasil, 2010. 640 p. ISBN 8588639181. Citado na página 72.

L. S. Shapley. A Value for N-Person Game. *Annals of Mathematics Studies, Princeton University*, v. 2, p. 307–317, 1953. Citado na página 42.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York, NY, USA: ACM, 2010. (Hotnets-IX), p. 19:1–19:6. ISBN 978-1-4503-0409-2. Disponível em: <http://doi.acm.org/10.1145/1868447.1868466>. Citado 2 vezes nas páginas 58 e 59.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 493–512, First 2014. ISSN 1553-877X. Citado 7 vezes nas páginas 19, 23, 32, 33, 34, 35 e 36.

LEVIN, D. et al. Logically centralized?: State distribution trade-offs in software defined networks. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 1–6. ISBN 978-1-4503-1477-0. Disponível em: <http://doi.acm.org/10.1145/2342441.2342443>. Citado na página 23.

LUN, J.; GRACE, D. Software defined network for multi-tenancy resource sharing in backhaul networks. In: *Wireless Communications and Networking Conference Workshops (WCNCW), 2015 IEEE*. [S.l.: s.n.], 2015. p. 1–5. Citado na página 19.

MACEDO, D. F. et al. Programmable networks x2014;from software-defined radio to software-defined networking. *IEEE Communications Surveys Tutorials*, v. 17, n. 2, p. 1102–1125, Secondquarter 2015. ISSN 1553-877X. Citado 10 vezes nas páginas 22, 23, 24, 26, 27, 29, 31, 34, 35 e 36.

MANDRU, N.; BAVIRISETTI, D.; KHARA, S. An Efficient Vertical Handoff Technique for Two-Tier Heterogeneous Networks. *ijpttjournal.org*, v. 2, p. 50–55, 2012. Disponível em: <http://www.ijpttjournal.org/volume-2/issue-2/IJPTT-V2I2P402.pdf>. Citado na página 41.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>. Citado 6 vezes nas páginas 8, 23, 30, 32, 33 e 34.

MOHAN, P. M.; DIVAKARAN, D. M.; GURUSAMY, M. Proportional bandwidth sharing using bayesian inference in sdn-based data centers. In: *2016 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2016. p. 1–6. Citado na página 20.

MORALES, L. V.; MURILLO, A. F.; RUEDA, S. J. Extending the floodlight controller. In: *2015 IEEE 14th International Symposium on Network Computing and Applications*. [S.l.: s.n.], 2015. p. 126–133. Citado na página 59.

MORZHOV, S.; ALEKSEEV, I.; NIKITINSKIY, M. Firewall application for floodlight sdn controller. In: *2016 International Siberian Conference on Control and Communications (SIBCON)*. [S.l.: s.n.], 2016. p. 1–5. Citado na página 50.

NAKAGAWA, Y.; HYOUDOU, K.; SHIMIZU, T. A management method of ip multicast in overlay networks using openflow. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 91–96. ISBN 978-1-4503-1477-0. Disponível em: <http://doi.acm.org/10.1145/2342441.2342460>. Citado na página 19.

NASH, J. The Bargaining Problem. *Econometrica*, v. 18, n. 2, p. 155–162, 1950. Citado na página 53.

NUNES, B. et al. Software-defined-networking-enabled capacity sharing in user-centric networks. *Communications Magazine, IEEE*, v. 52, n. 9, p. 28–36, September 2014. ISSN 0163-6804. Citado 2 vezes nas páginas 23 e 26.

NUNES, B. A. A. et al. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, v. 16, n. 3, p. 1617–1634, Third 2014. ISSN 1553-877X. Citado 9 vezes nas páginas 8, 19, 23, 24, 26, 27, 36, 37 e 38.

OPENFLOW Switch Specication. In: . Open, Networking Foundation, 2013. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf>. Citado na página 66.

ORACLE. *Java Java*. 2017. Disponível em: <https://www.oracle.com/br/java/index.html>. Citado na página 50.

P. Ameigeiras, J. Wigard, P. M. Performance of the m-lwdf scheduling algorithm for streaming services in hsdpa. *IEEE Trans. Veh. Technol. Conf.*, v. 2, p. 999–1003, 2004. Citado na página 42.

PATEL, A. et al. Survivable virtual infrastructure mapping with shared protection in transport software defined networks (t-sdns). In: *Optical Fibre Technology, 2014 OptoElectronics and Communication Conference and Australian Conference on*. [S.l.: s.n.], 2014. p. 679–681. Citado na página 19.

R. Basukala, H. Mohd Ramli, K. S. Performance analysis of EXP/PF and M-LWDF in downlink 3GPP TLE system. *IEEE F. Asian Himalayas Conf.*, p. 1–5, 2009. Citado na página 42.

REALE, R. F.; NETO, W. C. P.; MARTINS, J. S. B. Modelo de Alocação de Banda com Compartilhamento Oportunista entre Classes de Tráfego e Aplicações em Redes Multiserviço. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 163–176, 2011. Citado na página 40.

SALGUEIRO, E. Tese de Doutorado, *Alocação de Recursos com Justiça: Uma Aplicação de Jogos Cooperativos em Redes de Computadores.* 2009. 205 p. Citado 6 vezes nas páginas 19, 20, 41, 42, 48 e 49.

SEZER, S. et al. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, v. 51, n. 7, p. 36–43, July 2013. ISSN 0163-6804. Citado 5 vezes nas páginas 8, 23, 25, 29 e 31.

SHAN, T.; YANG, O. W. W.; MEMBER, S. Bandwidth Management for Supporting Traffic Engineering. *IEEE Transactions on Parallel and Distributed Systems*, v. 18, n. 9, p. 1320–1331, 2007. Citado na página 42.

SHIN, K. et al. Multimedia Service Discrimination Based on Fair Resource Allocation Using Bargaining Solutions. *ETRI Journal*, v. 34, n. 3, p. 341–351, jun 2012. ISSN 1225-6463. Disponível em: <http://ocean.kisti.re.kr/downfile/volume/etri/HJTODO/2012/v34n3/HJTODO_2012_v34n3_341.pdf>. Citado na página 41.

SHNAYDER, V. et al. Truthful prioritization for dynamic bandwidth sharing. In: *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. New York, NY, USA: ACM, 2014. (MobiHoc '14), p. 235–244. ISBN 978-1-4503-2620-9. Disponível em: <http://doi.acm.org/10.1145/2632951.2632956>. Citado na página 20.

STALLINGS, W. Software-defined networks and OpenFlow. *The Internet Protocol Journal*, Cisco, v. 16, n. 1, 2013. Citado na página 72.

SULIMAN, I. et al. Radio resource allocation in heterogeneous wireless networks using cooperative games. In: *Proc. Nordic Radio Symposium*. [s.n.], 2004. Disponível em: <http://www.ee.oulu.fi/~jannel/PDFs/suliman_NRS04.pdf>. Citado na página 41.

SURESH, L. et al. Towards programmable enterprise wlans with odin. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 115–120. ISBN 978-1-4503-1477-0. Disponível em: <http://doi.acm.org/10.1145/2342441.2342465>. Citado na página 38.

TANENBAUM, A. S. *Redes de Computadores*. 5ª. ed. São Paulo: Pearson do Brasil, 2011. 563 p. ISBN 978-85-7605-924-0. Citado na página 72.

TENNENHOUSE, D. L. et al. A survey of active network research. *IEEE Communications Magazine*, v. 35, n. 1, p. 80–86, Jan 1997. ISSN 0163-6804. Citado na página 25.

TIAN, J. et al. Fairshare: Dynamic max-min fairness bandwidth allocation in datacenters. In: *2016 IEEE Trustcom/BigDataSE/ISPA*. [S.l.: s.n.], 2016. p. 1463–1470. Citado 2 vezes nas páginas 58 e 59.

TRESTIAN, R.; ORMOND, O.; MUNTEAN, G.-M. Reputation-based network selection mechanism using game theory. *Physical Communication*, Elsevier B.V., v. 4, n. 3, p. 156–171, sep 2011. ISSN 18744907. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1874490711000322>. Citado na página 41.

WAGNER, D. P. Congestion Policing Queues - A new approach to managing bandwidth sharing at bottlenecks. In: *Network and Service Management (CNSM), 2014 10th International Conference on*. [S.l.: s.n.], 2014. p. 206–211. Citado na página 20.

XAVIER, F. C. Dissertação de Mestrado, *Cognar - Um sistema para alocação dinâmica de recursos baseado em técnicas de Inteligência Artificial.* 2012. 106 p. Citado na página 49.

XIA, W. et al. A survey on software-defined networking. *IEEE Communications Surveys Tutorials*, v. 17, n. 1, p. 27–51, Firstquarter 2015. ISSN 1553-877X. Citado 3 vezes nas páginas 34, 38 e 39.

YAN, B. et al. Cab: A reactive wildcard rule caching system for software-defined networks. In: *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. New York, NY, USA: ACM, 2014. (HotSDN '14), p. 163–168. ISBN 978-1-4503-2989-7. Disponível em: <http://doi.acm.org/10.1145/2620728.2620732>. Citado na página 43.

ZILBERMAN, N. et al. Reconfigurable network systems and software-defined networking. *Proceedings of the IEEE*, v. 103, n. 7, p. 1102–1124, July 2015. ISSN 0018-9219. Citado na página 23.

ZOPE, N.; PAWAR, S.; SAQUIB, Z. Firewall and load balancing as an application of sdn. In: *2016 Conference on Advances in Signal Processing (CASP)*. [S.l.: s.n.], 2016. p. 354–359. Citado na página 59.