



UNIVERSIDADE FEDERAL DE CATALÃO (UFCAT)
INSTITUTO DE MATEMÁTICA E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E OTIMIZAÇÃO



Jaime Ribeiro Júnior

**LÓGICA FUZZY APLICADA AO GERENCIAMENTO DE REDES
DEFINIDAS POR SOFTWARE**

DISSERTAÇÃO DE MESTRADO

CATALÃO – GO, 2023



UNIVERSIDADE FEDERAL DE GOIÁS
UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a [Lei 9.610/98](#), o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

☒ Dissertação ☐ Tese ☐ Outro*: _____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Jaime Ribeiro Júnior

3. Título do trabalho

LÓGICA FUZZY APLICADA AO GERENCIAMENTO DE REDES DEFINIDAS POR SOFTWARE

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento ☒ SIM ☐ NÃO¹

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:

a) consulta ao(a) autor(a) e ao(a) orientador(a);

b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.

O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.



Documento assinado eletronicamente por **Vaston Goncalves Da Costa, Orientador**, em 09/10/2023, às 12:27, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **JAIME RIBEIRO JÚNIOR, Discente**, em 09/10/2023, às 12:38, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4110624** e o código CRC **AD37AABB**.

JAIME RIBEIRO JÚNIOR

LÓGICA FUZZY APLICADA AO GERENCIAMENTO DE REDES
DEFINIDAS POR SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Otimização, do Instituto de Matemática e Tecnologia, da Universidade Federal de Catalão (UFCAT), como requisito parcial para a obtenção do título de Mestre em Modelagem e Otimização. Área de Concentração: Modelagem e Otimização. Linha de Pesquisa: Modelagem e Otimização.

Orientador:
Dr. Vaston Gonçalves da Costa

CATALÃO – GO

2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFCAT.

Ribeiro Júnior, Jaime

Lógica Fuzzy aplicada ao Gerenciamento de Redes Definidas Por
Software / Jaime Ribeiro Júnior. - 2023.
91, XCI f.

Orientador: Prof. Vaston Gonçalves da Costa.

Dissertação (Mestrado) - Universidade Federal de Catalão, Instituto
de Matemática e Tecnologia, Catalão, Programa de Pós-Graduação
em Modelagem e Otimização, Catalão, 2023.

Bibliografia. Apêndice.

Inclui siglas, gráfico, tabelas, algoritmos, lista de figuras, lista de
tabelas.

1. SDN. 2. Controle de Banda. 3. Inteligência Artificial. 4. Lógica
Fuzzy. I. Costa, Vaston Gonçalves da, orient. II. Título.

CDU 004



UNIVERSIDADE FEDERAL DE GOIÁS

UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº 23 da sessão de Defesa de Dissertação de **Jaime Ribeiro Júnior** que confere o título de Mestre(a) em **Modelagem e Otimização**, na área de concentração em **Modelagem e Otimização**.

Aos **tres dias do mes de outubro**, a partir da **10 horas**, via Web Conferencia <https://meet.google.com/afd-yjxs-mtf>, reuniram-se os componentes da banca examinadora, professores **Dr. Vaston Gonçalves da Costa (PPGMO/UFCAT), (orientador), Dr. Marcelo Henrique Stoppa (PPGMO / UFCAT), membro titular interno e Dr. Anderson Cavalcante Gonçalves (UEG), membro titular externo**, para, em sessão pública, procederem a avaliação da Dissertação intitulada **“LÓGICA FUZZY APLICADA AO GERENCIAMENTO DE REDES DEFINIDAS POR SOFTWARE”**, de autoria de **Jaime Ribeiro Júnior**, discente do Programa de Pós-graduação em Modelagem e Otimização – PPGMO, da Universidade Federal de Catalão - UFCAT. A sessão foi aberta pelo presidente, que fez a apresentação formal dos membros da banca. Em seguida, a palavra foi concedida ao discente que, em 35 min procedeu a apresentação. Terminada a apresentação, cada membro da banca arguiu o examinando. Terminada a fase de arguição, procedeu-se a avaliação da Dissertação, que foi considerado: **(X) Aprovado ou () Reprovado**. Cumpridas as formalidades de pauta, a presidência da mesa encerrou a sessão e, para constar, lavrou-se a presente ata que, depois de lida e aprovada, segue assinada pelos membros da banca examinadora.

TÍTULO SUGERIDO PELA BANCA



Documento assinado eletronicamente por **Vaston Goncalves Da Costa, Professor do Magistério Superior**, em 03/10/2023, às 18:49, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Marcelo Henrique Stoppa, Professor do Magistério Superior**, em 03/10/2023, às 23:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Anderson Cavalcante Goncalves, Usuário Externo**, em 09/10/2023, às 17:02, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4095678** e o código CRC **FA94E325**.

Agradecimentos

A expressão de minha gratidão se estende a todos aqueles que trilharam este trajeto ao meu lado, incutindo em mim a força e a inspiração necessárias. Um profundo agradecimento é dirigido aos meus familiares, àqueles que constituem os pilares da minha jornada: meus amados pai e mãe, Percival e Clarete, cujo apoio pavimentou o caminho até este ponto, minha amada noiva Mariana, cujo estímulo e incentivo foram inestimáveis, e minha sogra Vânia e demais pessoas, cujas palavras de estímulo e preces me acompanharam.

Minha gratidão também é direcionada aos ilustres mestres que enriqueceram este percurso e que honraram com sua presença na minha banca de defesa.

Ao professor Dr. Vaston Gonçalves da Costa, que foi meu guia durante a especialização, e mais uma vez é meu Orientador, foi fundamental na elaboração deste trabalho.

Aos distinguidos membros da banca de avaliação, o Prof. Dr. Marcelo Henrique Stoppa e o Prof. Dr. Anderson Cavalcante Gonçalves, que gentilmente aceitaram a responsabilidade de avaliar meu trabalho. Ao Professor Dr. José dos Reis, cuja orientação na disciplina de Inteligência Artificial abriu os caminhos para a realização da parte prática deste trabalho e publicação do artigo, expresso minha gratidão. Suas orientações foram fundamentais para o desenvolvimento deste projeto.

Também desejo estender meu agradecimento aos estimados Professores Dr. Celso Vieira Abud e Dr. Romes Antônio Borges, cuja participação nessa jornada enriqueceu meu repertório de conhecimentos de maneira significativa. Suas valiosas contribuições foram fundamentais para o êxito deste empreendimento acadêmico.

Agradeço de forma sincera a todos vocês por compartilharem este capítulo da minha jornada. Com humildade e alegria, reconheço a importância de vocês em minha trajetória acadêmica e pessoal.

E, em destaque, não posso deixar de expressar meu agradecimento especial à minha filha, Sophia de A. Ribeiro, uma fonte constante de inspiração e motivação em todos os momentos desta jornada.

RESUMO

Este trabalho apresenta os resultados de uma pesquisa que utiliza a Lógica Fuzzy para investigar a alocação equitativa de largura de banda em redes, considerando diversos perfis de usuários. A Lógica Fuzzy oferece flexibilidade ao distribuir largura de banda, considerando variáveis difusas, como requisitos de largura de banda, prioridades de tráfego e padrões de uso. O objetivo é desenvolver aplicações para Redes Definidas por Software (SDN) que integrem a Lógica Fuzzy, melhorando a eficiência da alocação de recursos de rede em ambientes complexos. Essa pesquisa é motivada pela crescente necessidade de automação na gestão de redes, impulsionada pelo aumento da Internet das Coisas (IoT) e da demanda por conectividade. As SDNs oferecem uma abordagem promissora para essa automação, permitindo a separação do plano lógico do físico e a integração de aplicativos inteligentes. A alocação eficiente de largura de banda é vital para otimizar o desempenho da rede e a satisfação do usuário, tornando esse campo de pesquisa significativo. Além da catalogação de tecnologias SDN, revisão bibliográfica em IA com ênfase na Lógica Fuzzy e desenvolvimento de algoritmos inteligentes também são apresentados resultados em um protótipo desenvolvido em Python. A análise dos resultados avaliou a equidade e qualidade das alocações de largura de banda. O trabalho contribui para o campo de SDNs e IA, destacando como a Lógica Fuzzy pode melhorar a distribuição de largura de banda em redes de computadores. A automação e alocação eficiente de recursos são cruciais para atender às demandas das redes modernas, e esta pesquisa avança em direção a soluções inteligentes.

Palavras-chaves: SDN, Controle de Banda, Inteligência Artificial, Lógica Fuzzy.

ABSTRACT

This paper presents the results of a research study that utilizes Fuzzy Logic to investigate the equitable allocation of bandwidth in networks, considering various user profiles. Fuzzy Logic offers flexibility in bandwidth distribution, taking into account Fuzzy variables such as bandwidth requirements, traffic priorities, and usage patterns. The goal is to develop applications for Software-Defined Networks (SDNs) that incorporate Fuzzy Logic, thereby enhancing the efficiency of network resource allocation in complex environments. This research is motivated by the growing need for network management automation, driven by the rise of the Internet of Things (IoT) and the demand for connectivity. SDNs offer a promising approach to this automation, allowing for the separation of the logical and physical network planes and the integration of intelligent applications. Efficient bandwidth allocation is critical for optimizing network performance and user satisfaction, making this research field significant. In addition to cataloging SDN technologies, conducting a literature review on AI with an emphasis on Fuzzy Logic, and developing intelligent algorithms, this paper also presents results from a prototype implemented in Python. The analysis of these results evaluated the equity and quality of bandwidth allocations. This work contributes to the fields of SDNs and AI, highlighting how Fuzzy Logic can enhance bandwidth distribution in computer networks. Automation and efficient resource allocation are crucial to meet the demands of modern networks, and this research advances toward intelligent solutions.

Keywords: SDN, Data Control, Artificial Intelligence, Fuzzy Logic.

LISTA DE FIGURAS

Figura 2.1 – Arquitetura de Hardware SDN.	23
Figura 2.2 – Arquitetura de Software SDN.	24
Figura 2.3 – Comparação a partir do Conceito de virtualização de rede.	27
Figura 2.4 – Relação entre as áreas de pesquisa de I.A, Aprendizado de Máquina e Deep Learnig.	33
Figura 2.5 – Sistema Lógico Binário: $(A + B).(\neg A + C)$	34
Figura 2.6 – Tabelas de Operações Lógicas Booleanas	35
Figura 2.7 – Variável Nebulosa (linguística)	39
Figura 2.8 – Tipos de Função de Pertinência	41
Figura 2.9 – Modelo Simplificado de um Sistema Difuso	42
Figura 3.1 – Arquitetura CAARF-SDN.	52
Figura 4.1 – Funções de Pertinências do Protótipo do Sistema Fuzzy	67
Figura 4.2 – Diagrama de Caso de Uso do Sistema Protótipo	70
Figura 4.3 – Saídas do Experimento A	71
Figura 4.4 – Saídas do Experimento B	71
Figura 4.5 – Saídas do Experimento C	72
Figura 4.6 – Saídas do Experimento D	73

LISTA DE TABELAS

Tabela 2.1 – Operador sob conjunto difuso OR	43
Tabela 2.2 – Operador sob conjunto difuso E	43
Tabela 2.3 – Operador sob conjunto difuso NOT	43
Tabela 2.4 – Métodos de Acumulação	45
Tabela 2.5 – Principais Operações de defuzzificação	45

LISTA DE ABREVIATURAS E SIGLAS

BGP — Border Gateway Protocol

CAARF-SDN — Context-Aware Adaptive Routing Framework SDN

CAPES — Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

DDoS — Negação de Serviços Distribuídos

DoS — Negação de Serviços

GNV — Global Network View

IA — Inteligência Artificial

IaaS — Infrastructure as a Service

IP — Internet Protocol

LN — Lógica Nebulosa

LPA — Lógica Paraconsistente Anotada

NOS — Network Operating System

NV — Network Virtualization - virtualização de redes

OFP — Openflow Protocol

ONF — Open Network Foundation

OSPF — Open Shortest Path First

QoD — Qualidade de Dispositivo

QoE — Qualidade de Experiência

QoS — Qualidade de Serviço

RIP — Routing Information Protocol

SSL — Secure Socket Layer

T.I. — Tecnologia da Informação

TCP — Transmission Control Protocol

LISTA DE CÓDIGOS

Código 2.1 – Inferencia Max-Min	43
Código 2.2 – Tipificação das Regras TSK	44
Código 3.1 – Código Python análise métrica	62
Código 3.2 – Lógica Fuzzy e Mininet	63
Código A.1 – Instalar o Pacote	85
Código A.2 – Importar as Bibliotecas	85
Código A.3 – Conjuntos Fuzzy	85
Código A.4 – Criando as Funções de Pertinência	86
Código A.5 – Visualizando as funções de Pertinência	87
Código A.6 – Fuzzyficando os Dados	88
Código A.7 – Vizualizando dados Fuzzyficados	89
Código A.8 – Aplicando Regras	89
Código A.9 – Visualizando Regras aplicadas	90
Código A.10 – Inferencia	90
Código A.11 – Defuzzificnado Dados	91

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Justificativa	15
1.2	Objetivos da Pesquisa	16
1.3	Metodologia	17
2	REFERENCIAL TEÓRICO	19
2.1	A Rede Definida por Software - SDN	20
2.1.1	A Open Network Foundation	21
2.2	Arquitetura da SDN	23
2.2.1	O Flowvisor	26
2.2.2	O Openflow	27
2.2.3	As interfaces Norte e Sul	30
2.2.4	O Mininet - Arcabouço para implementação e testes práticos em SDN	31
2.3	Inteligencia Artificial e o Aprendizado de Máquina	32
2.4	A Lógica Nebulosa (Lógica Fuzzy)	33
2.4.1	Lógica binária vs lógica Nebulosa	34
2.4.2	Lógica Fuzzy	37
2.4.2.1	Fuzzificação	38
2.4.2.2	Variável linguística e variáveis numérica	38
2.4.2.3	Função de Pertinência	39
2.4.2.4	Base de Conhecimento	42
2.4.2.5	Inferência	43
2.4.2.6	Defuzzificação	45
3	TRABALHOS RELACIONADOS	47
3.1	Aplicação de funções de Pertinência na Lógica Fuzzy	47
3.2	Controle SDN e suas as aplicações em Redes de Computadores	50
3.3	Roteamento SDN e QoS	53
3.4	A SDN em soluções proprietárias	55
3.4.1	Redes Baseadas na Intenção	55
3.4.2	A solução Ubiquiti Unifi	56
3.4.3	Abordagem da VMware em SDN	58

3.5	Como Implementar Lógica Fuzzy utilizando Phyton e a Mininet	60
3.5.1	Implementação	61
3.5.2	Projetos públicos sob SDN e QoS	61
4	DETERMINAÇÃO DE LARGURA DE BANDA COM LÓGICA FUZZY	65
4.1	Saídas e Avaliação dos Resultados	70
5	CONCLUSÃO E TRABALHOS FUTUROS	74
REFERÊNCIAS		77
APÊNDICE A	CÓDIGO COLAB	85

Capítulo 1

INTRODUÇÃO

O estilo de vida da sociedade está a cada dia mais dependente da Internet, o que é evidente pelo grande número de usuários cadastrados em diversas mídias sociais e pelas consideráveis transações bancárias realizadas pela rede. Com a popularização das redes sem fio, os dispositivos portáteis pessoais, como *smartphones*, se tornaram mais comuns contribuindo para o aumento da quantidade de pessoas ativas na rede de computadores. Além disso, a indústria do entretenimento está gradualmente migrando para serviços online sob demanda (*streaming*) (LE MOS; PASTOR; OLIVEIRA, 2012).

Com a pandemia da Covid-19 e as políticas de restrição e isolamento social que se seguiram, muitas interações sociais passaram a ser realizadas pela Internet, como aulas, reuniões e shows. Aproximadamente 74% das transações bancárias realizadas por pessoas físicas em 2020 foram feitas por canais digitais como aplicativos bancários e internet *banking*.

Com a difusão das facilidades advindas do emprego de técnicas de inteligência artificial, é possível incluir um computador em qualquer eletrodoméstico e implantar sistemas baseados em algoritmos inteligentes. Isso é uma área de pesquisa em ascensão no mercado, denominada Internet das Coisas (IoT). O aumento crescente por pontos de conexão na Internet, potencializado pela IoT e pela popularização crescente dos meios de comunicação informatizados, demanda uma constante expansão física e lógica da rede (TELECOMUNICAÇÕES, 2021).

A ideia de separar o plano físico e lógico da rede tornando-os transparentes um ao outro permite que as redes possam ser controladas a partir de softwares independentes do hardware. Isso é possível através da técnica de virtualização das redes. Com isso, camadas de softwares de alto nível podem ser facilmente inseridos na rede com o intuito de controlá-la. Esses softwares podem fazer uso da inteligência artificial para identificar padrões, tomar decisões e escalonar valores de configuração automaticamente. Essas redes são denominadas Redes Definidas por Software (SDN).

A arquitetura SDN, cujo termo foi cunhado em 2005 com base em pesquisas da Uni-

versidade de Stanford, representa um paradigma de controle do hardware de infraestrutura por meio de software. A principal vantagem desse paradigma é a capacidade de gerenciar todos os componentes de hardware da rede de forma eficiente e altamente flexível por meio de softwares controladores. (UBIQUITI, 2018)

As SDN foram apresentadas como um novo tipo de solução para o problema de re-configuração, atualização, manutenção e gerência de redes de computadores. Uma SDN é basicamente dividida em Servidores de Controle, Servidor Proxy de Virtualização e dispositivos SDN. Neste trabalho, busca-se apresentar um software inteligente que auxilie na administração do controle de SDN.

Neste contexto foi realizado de um experimento que emprega a Lógica Fuzzy como abordagem para investigar a eficiência na alocação equitativa de largura de banda aos usuários de uma rede, considerando os diferentes perfis de cada usuário. Este experimento visa abordar uma distribuição de largura de banda que leve em conta as características individuais dos usuários, como requisitos de largura de banda, prioridades de tráfego e padrões de uso.

Este trabalho apresenta contribuições ao fornecer um recurso de estudo abordando tanto a teoria quanto a aplicação prática das tecnologias de Redes Definidas por Software (SDN) e Lógica Fuzzy. Além disso, também foi apresentado um algoritmo inteligente, capaz de estimar a largura de banda com base nos parâmetros de uma rede de computadores. Este algoritmo representa um protótipo escalável, pronto para futuras expansões e aplicação em ambientes de Redes Definidas por Software (SDN).

A construção de aplicações para Redes Definidas por Software (SDN) que empregam Lógica Fuzzy para modelar o problema de atribuição de largura de banda em redes de computadores é uma abordagem relevante para esta área. Esse tipo de aplicação pode proporcionar uma alocação mais eficiente e adaptativa de recursos de rede, levando em consideração a complexidade e as variáveis dinâmicas das redes modernas.

1.1 Justificativa

O emprego de Redes Definidas por Software (SDN) pode revolucionar a forma como as redes de computadores são mantidas em funcionamento (CISCO, 2021). Separando o plano de hardware do plano lógico no funcionamento das redes de computadores, cópias virtuais adicionais das redes podem ser entregues aos controladores para aplicação de teste de novas configurações (PINHEIRO, 2016). Tendo em vista a necessidade de mão de obra qualificada na área de gerência de redes de computadores, que está em alta e que os profissionais formados que atendem as expectativas não são suficientes, aplicações inteligentes que possam automatizar o processo de gerência são desejáveis (SALOMÃO, 2021).

O gerenciamento de redes de computadores de infraestrutura é uma tarefa constante e muitas vezes, a entrada de novos pontos de conexão na rede provoca uma nova configuração de distribuição de banda, de forma a otimizar a distribuição dos recursos de dados da rede. Ter aplicações que façam esse balanceamento automaticamente a partir da análise do tráfego da rede e das opções de prioridades designadas pelo administrador é um diferencial para a qualidade da conexão provida pela empresa a seus usuários. Além disso, os aplicativos inteligentes podem ser incorporados à rede a partir de sua execução em um servidor com um controlador SDN, sendo que a abstração do hardware promovida pelo Openflow permite que novas funcionalidades sejam implementadas independentemente dos softwares proprietários dos equipamentos (CISCO, 2021).

As Redes Definidas por Software (SDN) emergem como uma resposta às crescentes demandas por inovações na área de Redes de Computadores, especialmente quando associadas à Inteligência Artificial (IA) para o desenvolvimento de aplicações inteligentes. Esta abordagem dedica-se em oferecer uma análise da integração de SDN e IA, além de propor exemplos práticos para sua implementação. Essa interseção entre SDN, IA e suas aplicações práticas representam uma contribuição atual para o campo das redes de computadores, em especial SDN, abordando as necessidades emergentes desta área que está em constante evolução.

1.2 Objetivos da Pesquisa

O objetivo principal desta pesquisa é explorar a aplicação de técnicas de Inteligência Artificial (IA), em especial Lógica Fuzzy, no contexto de Redes Definidas por Software (SDN) com o propósito de desenvolver uma solução para o desafio da distribuição de largura de banda, com ênfase na garantia de Qualidade de Serviço (QoS) para os usuários da rede.

Especificamente, os objetivos detalhados desta pesquisa são os seguintes:

1. **Catálogo de Tecnologias SDN:** Realizar uma revisão das tecnologias SDN já existentes, identificando suas funcionalidades, capacidades e aplicações relevantes.
2. **Revisão Bibliográfica sobre Inteligência Artificial com foco na Lógica Fuzzy:** Conduzir uma revisão bibliográfica para identificar soluções de Inteligência Artificial adequadas para abordar o problema específico relacionado à distribuição de largura de banda em redes SDN.
3. **Desenvolvimento de Algoritmo Inteligente:** Construir algoritmos inteligentes personalizados para resolver desafios específicos, incluindo o controle de largura de banda, gerando estimativas de largura de banda utilizando Python e que futuramente podem ser incorporados em um controlador SDN para prover QoS.

4. **Elaborar um material para orientar novos pesquisadores na área de SDN:** Ainda hoje, notamos a carência de literatura didática significativa que aborde detalhadamente o universo das Redes Definidas por Software (SDN). Embora existam diversos trabalhos acadêmicos disponíveis, essas informações geralmente se encontram fragmentadas. Diante desse cenário, este trabalho se apresenta como uma fonte de pesquisa, com a finalidade de proporcionar ao leitor um acesso à teoria e à prática relacionadas a esse campo em constante evolução.
5. **Apresentar os Resultados Obtidos:** Apresentar os resultados obtidos no protótipo criado em Python para visualização das estimativas de largura de banda geradas e analisar sua equidade e qualidade.

1.3 Metodologia

A pesquisa foi conduzida em uma série de etapas sequenciais, com o objetivo de desenvolver e implementar soluções baseadas em Redes Definidas por Software (SDN) e inteligência artificial para abordar o problema proposto. As etapas são descritas detalhadamente a seguir:

Inicialmente, realizou-se uma pesquisa para catalogar as tecnologias SDN já existentes no mercado e na literatura acadêmica. O objetivo era obter uma compreensão abrangente das soluções disponíveis e suas capacidades.

Uma revisão bibliográfica foi realizada para identificar soluções de inteligência artificial adequadas para integrar ao ambiente SDN. Isso incluiu a análise de algoritmos de aprendizado de máquina, técnicas de processamento de dados e métodos de automação.

Com base nas descobertas da revisão bibliográfica, a Lógica Fuzzy emergiu como uma abordagem apropriada para resolver o problema proposto. Esta solução foi implementada com sucesso em um protótipo, resultando em uma coletânea de dados passíveis de análise, com vistas a avaliar a viabilidade de sua aplicação no controle de largura de banda em sistemas de rede.

A seleção da Lógica Fuzzy como metodologia para abordar o problema foi embasada em sua capacidade de lidar com a incerteza e a imprecisão inerentes a ambientes de redes de computadores. A natureza fluida e dinâmica das redes modernas frequentemente desafiam os métodos tradicionais de alocação de largura de banda. A Lógica Fuzzy, com sua capacidade de considerar variáveis difusas e adaptar-se a condições em constante mudança, mostrou-se promissora como uma abordagem mais flexível e sensível às demandas reais das redes.

Nesse experimento, a Lógica Fuzzy foi aplicada da seguinte maneira:

- **Modelagem do Problema:** A modelagem do problema envolve a definição de variáveis de entrada (como demanda de largura de banda, características dos aplicativos e tráfego de rede) e variáveis de saída (quantidade de largura de banda no contexto da velocidade de conexão em bits por segundo) usando conceitos de Lógica Fuzzy.
- **Regras Fuzzy:** A criação de regras Fuzzy que relacionam as variáveis de entrada às variáveis de saída é fundamental. Essas regras refletem o conhecimento especializado sobre como atribuir largura de banda com base nas condições da rede e nos requisitos dos aplicativos que são refletidas na função de pertinência.
 - **Conjuntos Fuzzy:** Define conjuntos Fuzzy que descrevem as características das variáveis de entrada e saída, como "baixa demanda de largura de banda", "alta congestão da rede" e "prioridade alta do aplicativo".
- **Inferência Fuzzy:** Utilizada para calcular a largura de banda alocada com base nas regras definidas e nas condições da rede em tempo real.
- **Ambiente de Desenvolvimento Python:** Foi utilizado um ambiente de desenvolvimento Python no google Colab, para gerenciar dependências e pacotes, e realizar os experimentos.
- **Bibliotecas de Lógica Fuzzy:** Foi utilizado a biblioteca de Lógica Fuzzy scikit-fuzzy, para criar os conjuntos Fuzzy, definir regras e realizar a inferência Fuzzy.
- **Coleta de Dados:** Foram coletados os dados gerados pelos estudos de caso, gerados pelo modelo Fuzzy com entradas a partir das informações sobre usuários, demanda de largura de banda, estado da rede, prioridades de tráfego, etc.
- **Resultados e Conclusões:** Análise dos resultados dos experimentos e conclusões sobre a eficácia da abordagem de lógica Fuzzy.

A implementação dessa solução em um protótipo permitiu gerar dados tangíveis que agora estão sujeitos a análise que visa avaliar a eficácia da Lógica Fuzzy na alocação de largura de banda.

Essa metodologia guiou o desenvolvimento e a implementação das soluções de SDN e inteligência artificial para abordar o problema proposto, possibilitando uma abordagem sistemática ao longo do processo de pesquisa.

Capítulo 2

Referencial Teórico

As redes de computadores estão intrinsecamente ligadas ao estilo de vida atual da maior parte da população mundial. Do bom dia dado a um amigo pelo WhatsApp, ao acesso feito no caixa eletrônico bancário, à compra paga por cartão de crédito ao tráfego de dados econômicos, às transações financeiras mundiais todos dependem das redes de computadores (BRETÃO, 2020). Corporações digitais como Google e Facebook se consolidaram devido a rede mundial de computadores.

Manter a Internet em pleno funcionamento é um trabalho árduo, uma interrupção de comunicação pode representar um grande prejuízo financeiro. A rede de responsabilidade privada, porém de proporções geográficas grandes, pode em caso de falha gerar um prejuízo financeiro bilionário, como foi o caso da empresa Facebook. Que, no dia 04 de outubro de 2021, teve um prejuízo de cerca de US \$6 bilhões em algumas horas, devido a um problema de alteração de rotas entre os roteadores de backbone - que coordenam o tráfego- e os data-centers (BULHÕES, 2021) (BRITO, 2021).

Não é incomum falhas como a citada anteriormente, as empresas que dependem das redes de computadores, como provedores de Internet, varejistas online, serviços por aplicativos, redes sociais, caixas postais de e-mail, redes bancárias, e vários outros, estão sujeitas a falhas em suas redes. A gerência, manutenção e proteção dessas redes de computadores não é trivial. É difícil encontrar profissionais especialistas nessa área, é uma área de conhecimento que exige experiência teórica e prática em iguais proporções para se ser um profissional de ponta constantemente atualizado. Estima-se que até 2024 a área da T.I. (Tecnologia da Informação) terá um deficit de cerca de 420 mil profissionais (SALOMÃO, 2021).

Com a popularização da Internet das Coisas (IoT) cada dia mais equipamentos vêm sendo incorporados as redes de computadores para os mais diferentes fins a partir de eletrodomésticos, controladoras industriais, controle de trafego de veículos, veículos automotores e vários outros. A IoT é o termo designado para uma tecnologia que possibilite que algo que antes não era usual, agora se conecte a Internet. Essa nova e crescente demanda de conexão

exigirá cada vez mais da infraestrutura das redes de computadores aumentando a demanda por tecnologias cada vez mais rápidas em um espaço de tempo cada vez menor, aumentando cada vez mais a demanda por mão de obra qualificada ([TELECOMUNICAÇÕES, 2021](#)).

A pilha de protocolo TCP/IP é fundamental para o funcionamento da rede de computadores. Com a crescente demanda, houve uma necessidade de mudança estrutural nos protocolos de redes de computadores. A tecnologia da pilha de protocolos TCP/IP é utilizada como padrão nas redes de computadores mundiais devido à sua simplicidade e flexibilidade. Isso tornou possível a interconexão das redes de diferentes nações. O protocolo de internet (IP) é um dos principais protocolos desta pilha, e atualmente existem duas versões em uso, o IPv4 e o IPv6. O IPv4 está sendo gradualmente substituído porque não pode mais fornecer endereços de IP públicos suficientes para atender à crescente demanda de equipamentos que precisam desse tipo de endereço ([TANENBAUM; WETHERALL, 2021](#)).

No entanto, a arquitetura atual tem seus problemas. A descentralização do controle de rotas de dados torna difícil identificar e solucionar problemas em tempo hábil. Além disso, a padronização da pilha de protocolos TCP/IP pode limitar a criação de novas tecnologias e aplicações.

Uma solução apresentada neste contexto são as Redes Definidas por Software (SDN). Com a SDN, o controle e gerenciamento das redes são centralizados e automatizados por meio de softwares inteligentes. Controladores centralizados tomam decisões de comutação e roteamento de dados, permitindo que a atualização de protocolos e arquiteturas lógicas seja muitas vezes limitada à atualização das configurações do software centralizado, simplificando a demanda de trabalho ([CAMPOS; MARTINS, 2017](#)).

Hoje, tudo está conectado à rede de computadores - usuários, dispositivos, aplicações, matrizes e filiais, aplicações em nuvem, continentes inteiros. A rede deve estar preparada para se adaptar, proteger e se recuperar de falhas. A rede também deve fornecer informações constantemente para um sistema que possibilite que tudo isso seja feito de forma automatizada. A SDN possibilita a automação na capacidade de gerenciamento de data centers, redes de longa distância e seus usuários ([CISCO, 2021](#)).

2.1 A Rede Definida por Software - SDN

De acordo com a *Networking Foundation*, SDN é uma alternativa que busca simplificar o controle de equipamentos de diferentes fabricantes, representando um novo paradigma nas redes de computadores. A dinamicidade e flexibilidade da SDN permitem que enlaces estáticos possam ser transformados em enlaces que podem ser alterados com a mudança de configurações em um controle centralizado. ([OPEN NETWORKING FOUNDATION, 2012](#)).

A virtualização de redes (NV) desempenha um papel complementar e significativo na abordagem da SDN. A combinação dessas tecnologias resulta nas chamadas Redes Virtuais Definidas por Software (vSDNs). A virtualização possibilita que os recursos da rede sejam compartilhados por diferentes controladores de software, cada um recebendo uma instância virtual da rede (redes virtuais ou fatias), onde os recursos compartilhados podem ser nós ou enlaces. Esse compartilhamento de recursos permite a geração de topologias dinâmicas. Essa abordagem é baseada em uma infraestrutura que também atua como serviço, conhecido como IaaS (Infrastructure as a Service) (KELLER; REXFORD, 2010). (BLENK *et al.*, 2016) afirma que a virtualização é uma importante complementação da SDN, formando a base para uma abordagem ainda mais flexível e dinâmica para o gerenciamento de redes.

Com a SDN é possível separar o plano lógico de comutadores e roteadores de seu hardware, movendo o controle lógico e de estado para um componente de software centralizado, separando o plano de dados do plano de controle (PINHEIRO, 2016). Esse software centralizado tendo em sua base de dados as informações da rede pode ser fazer uso de aprendizado de máquina e prover configurações automáticas em tempo real. Os setores do mercado presentes no contexto das redes de computadores, como provedores de Internet, desenvolvedores de soluções de hardware, provedores de serviços na nuvem tem interesse na SDN, portanto a ela não é somente de interesse acadêmico, é também de interesse do mercado (JAIN; PAUL, 2013).

O SDN provê aos arquitetos de redes liberdade para modelar o plano de controle, testá-lo, sem afetar o plano de dados. Com a separação dos planos de controle e dados é possível projetar a rede e pensar em um controle de maneira centralizada, olhando a visão geral da rede e não apenas para os enlaces e rotas específicos e adjacentes como no modelo distribuído, esse contexto de visão global é conhecido por GNV (*Global Network View* - Visão Global da Rede de Computadores) (LEVIN *et al.*, 2012).

O princípio fundamental das SDN é a sua capacidade de programabilidade e a manipulação de pacotes que se baseia no conceito de fluxos, este conceito de fluxos é baseado na sequência de pacotes que compartilharão atributos de valores definidos. A constituição de um fluxo é definida por funções dos recursos administrados pelas interfaces de programação. A *Open Network Foundation* - ONF é a empresa que vem normatizando e incentivando o desenvolvimento aberto das SDN. ((GUEDES *et al.*, 2012)

2.1.1 A Open Network Foundation

A ONF é uma organização apoiada por grandes empresas de comunicação e tecnologia como, por exemplo, Facebook, Cisco, Google, Intel e Huawei. Ela lidera os esforços na normatização do modelo de SDN que vem sendo desenvolvido, e aos poucos implementado. Essa organização visa principalmente desenvolver e atualizar o protocolo Openflow. (VOLPATO *et al.*, 2017)(VOLPATO, 2017)

A ONF tem o objetivo de popularizar as redes definidas por software difundindo e mantendo seus padrões de arquitetura, essa mesma organização também é responsável por criar e manter não apenas os padrões de arquitetura SDN mas também o *Openflow* e outros protocolos. (BARBOZA, 2019)

De acordo com (SOUZA, 2019), a comunicação em uma SDN é realizada através de APIs, que permitem a troca de informações e interfaces de comunicação. As APIs são divididas em duas partes: a API Sul (*Southbound*), responsável pela interface entre o plano de controle e o plano de dados físico; e a API Norte (*Northbound*), localizada entre as camadas de controle e aplicação, responsável pela comunicação com as aplicações.

Essa arquitetura promove a separação dos planos de dados e plano de controle, oferece programabilidade de rede por meio de aplicações e infra estrutura específica, visa trazer uma maior flexibilidade e facilidade na sua administração por disponibilizar de dispositivos com tecnologia avançada flexíveis e fáceis de gerenciar. (MOLINOS *et al.*, 2022)

A definição de uma interface clara e bem estruturada é crucial para o controle do encaminhamento de pacotes em comutadores de rede utilizando o protocolo Openflow. A API do comutador permite que o software inspecione, defina e modifique a tabela de roteamento. O software de controle é uma aplicação monolítica baseada em um controlador de aplicação geral e inclui aplicações específicas para o funcionamento da rede SDN. Adicionalmente, é possível utilizar um divisor de visões para alocar as aplicações em diferentes controladores. (GUEDES *et al.*, 2012)

A arquitetura proposta pela Open Networking Foundation (ONF) é baseada na separação da infraestrutura em três camadas: Camada de Aplicação, Camada de Controle e Camada de Infraestrutura.

A *Camada de Aplicação* é composta por aplicações escritas em linguagens de programação e interpretadas pelos controladores da rede, que definem as regras de encaminhamento de pacotes e se comunicam com a Camada de Controle por meio da interface Northbound, uma API de software bem definida.

A *Camada de Controle* abriga o controlador SDN, que é equipado com um software do tipo Network Operating System (NOS), responsável por controlar e definir as regras de encaminhamento de pacotes. Ao contrário das redes tradicionais, que possuem sistemas especializados para cada função em cada equipamento, na rede SDN, o sistema operacional é centralizado e único para toda a rede, controlando todas as ações da rede de forma centralizada e enviando comandos seguindo os parâmetros do Protocolo Openflow.

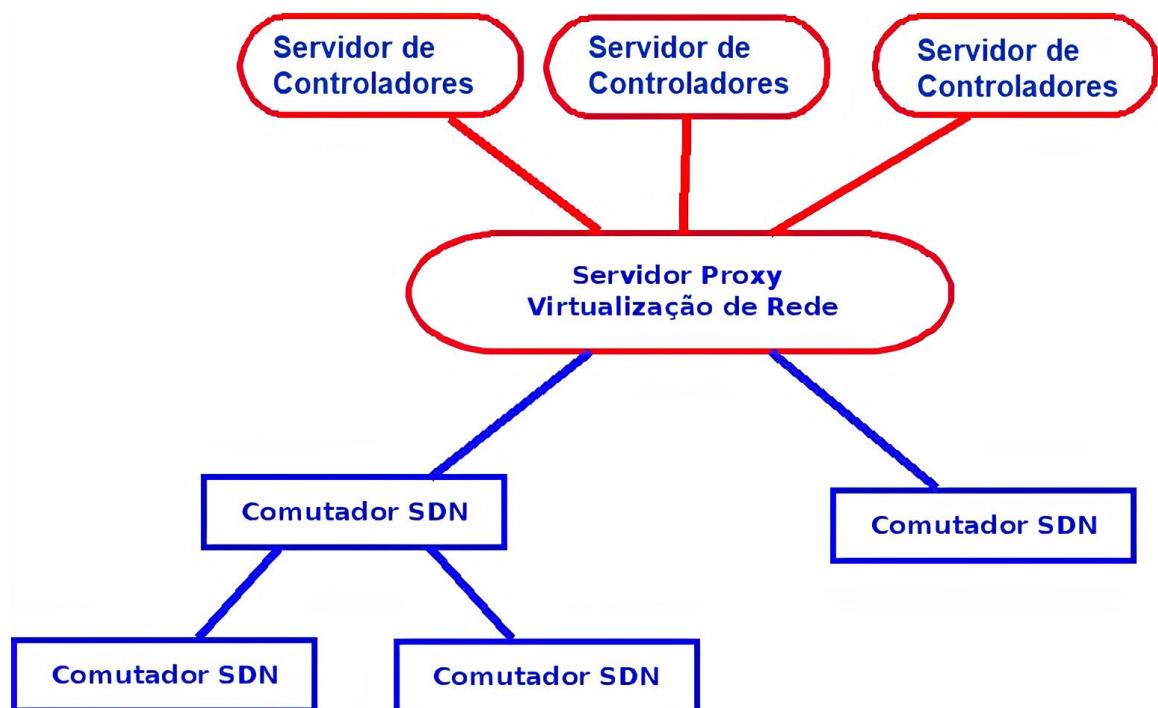
Por fim, a *Camada de Infraestrutura* é responsável pelo encaminhamento de pacotes e é composta por equipamentos como switches(comutadores) dotados do protocolo Openflow, que permite o controle SDN. Todas as decisões sobre o comportamento da rede são tomadas pelo controlador e são executadas na rede por meio da interface Sul (*Southbound*),

uma interface de comunicação localizada entre as camadas de controle e infraestrutura, que mantém a comunicação entre essas camadas e os equipamentos de encaminhamento de pacotes. (BARBOZA, 2019)

2.2 Arquitetura da SDN

O Hardware SDN requer equipamentos específicos para o seu funcionamento, como servidores proxy e comutadores, que diferem dos equipamentos tradicionais. É importante que esses equipamentos tenham características próprias para suportar a arquitetura do SDN (PINHEIRO *et al.*, 2013). Na figura 2.1, é possível visualizar um esquema básico da arquitetura de infraestrutura do Hardware SDN.

Figura 2.1 – Arquitetura de Hardware SDN.



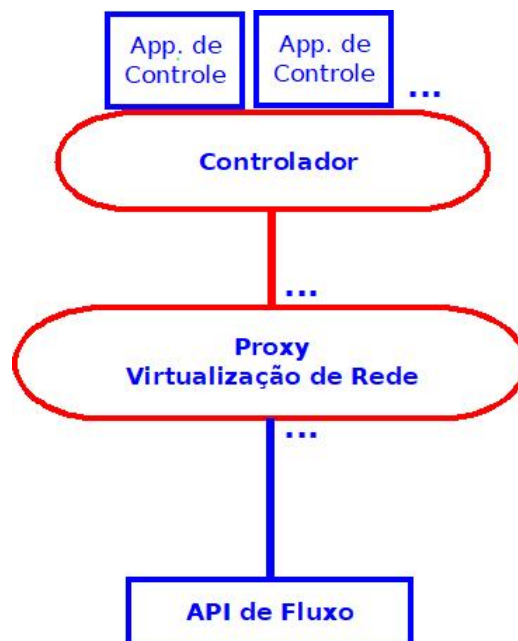
Fonte: (PINHEIRO *et al.*, 2013).

Os *Servidores de Controladores* executam os programas controladores da rede e podem incluir sistemas inteligentes. O *Servidor Proxy de Virtualização da Rede* é um servidor proxy que abstrai a rede física em redes virtuais e repassa instâncias virtuais da rede para cada controlador com permissões específicas, evitando conflitos de configuração. Juntamente com os *Servidores de Controle*, são considerados o cérebro da rede. Os *Comutadores SDN* são responsáveis pela comutação do fluxo da rede e diferem dos *switches* convencionais porque as decisões de controle são repassadas pelo controlador central. Por essa razão, um bom comutador SDN pode custar menos que um comutador tradicional de qualidade.

No entanto, uma desvantagem dessa arquitetura é que ela é completamente dependente dos servidores centralizados, diferentemente da rede tradicional. Isso torna esses servidores um ponto crítico da infraestrutura global da rede, o que significa que, em caso de falha dos servidores SDN centralizadores ou de sua rede, toda a rede ficará comprometida. Portanto, é essencial reforçar a atenção à segurança, integridade e disponibilidade da rede SDN.

Na arquitetura SDN, o comportamento da rede é definido principalmente pelos seus elementos de software centralizados. Esses elementos incluem o controlador de fluxo, a camada de virtualização, o software controlador e os aplicativos que são executados sobre os controladores. A Figura 2.2 mostra um diagrama dos elementos de software responsáveis pelo funcionamento da SDN, de acordo com (PINHEIRO *et al.*, 2013).

Figura 2.2 – Arquitetura de Software SDN.



Fonte: (PINHEIRO *et al.*, 2013).

De acordo com Pinheiro (PINHEIRO, 2016), a SDN é composta por diversos elementos de software, incluindo o *Controlador de Fluxos*, a *Camada de Virtualização*, o *Controlador* e os *Aplicativos de Controle*.

O *Controlador de Fluxos*, presente nos comutadores SDN, gerencia o fluxo de dados e é controlado por elementos externos através de diferentes APIs, sendo a API Openflow a mais conhecida.

A *Camada de Virtualização* é responsável por interligar todos os elementos da SDN e permitir a virtualização da rede, podendo ser utilizada com diferentes APIs, sendo a Flowvisor a mais conhecida.

O *Controlador*, também conhecido como *Network Operating System* (NOS) , é o software responsável por fornecer uma interface de programação para implementação dos aplicativos de controle, além de oferecer as funções tradicionais dos comutadores. Existem diversas implementações de Controladores SDN, como NOX, POX e Beacon.

Os *Aplicativos de Controle* são as aplicações criadas para definir o comportamento da rede, e podem utilizar APIs consolidadas para experimentar aplicações inteligentes e definir o comportamento da rede automaticamente.

O controlador em uma estrutura SDN funciona como o cérebro da rede ele fornece uma visão centralizada da rede aos administradores que podem ditar regras aos sistemas subjacentes ,essas regras ditam como os equipamentos devem funcionar em relação ao plano de encaminhamento e como deve ser feito o tráfego de dados pela rede. (TEIXEIRA, 2018)

Um controlador SDN centraliza todo o plano de controle e todas as regras de comunicação e de controle da infra estrutura da rede de dados , assim mantém o fluxo de comunicação frequente entre os dispositivos como switches, através do protocolo OpenFlow mantendo a troca de pacotes. Os controladores podem ser múltiplos na rede e são centralizados e dedicados. Essa abordagem de múltiplos controladores é útil, pois pode evitar falhas no sistema da rede devido a falhas em um componente centralizado. A comunicação entre o plano de controle e o plano de dados é feita por meio de interfaces. (SCHWEITZER; FERRARI; SHINODA, 2018)

De acordo com (FARES, 2021) o NOX, é um NOS , este sistema faz toda a interconexão dos equipamentos programáveis de rede, de forma centralizada, além de absorver a camada física adjacente toma as decisões acerca do envio de pacotes de dados, gerencia serviços e fluxos de entradas, podendo se encontrar no mesmo ambiente da rede ou em ambiente remoto instalado no controlador, a sua principal função é dar funcionalidade ao plano de dados e integrar os aplicativos SDN que ajudam a definir as redes.

De acordo com (FERNANDES; CARMO; FERNANDES, 2015) através das abstrações na SDN se destaca às características:

- O roteamento é feito com flexibilidade e monitorado por software.
- O Protocolo que opera de forma centralizada também opera nas camadas inferiores da rede.
- São os controladores que mapeiam os dispositivos de rede e fazem a configuração desses dispositivos.
- Podem gerar especificações de funcionamento de forma automática enviando configurações de acordo com as características físicas dos dispositivos.

- É possível uma abstração total da rede e de suas funções para uma Cloud (servidor remoto).

As SDN são inovações tecnológicas projetadas para tornar as redes de computadores mais flexíveis e ágeis. A separação dos planos de controle e de dados permite a facilidade de projeto, gerenciamento e monitoramento da rede, inclusive de forma remota. O plano de controle é programável e independente da infraestrutura de rede.

As SDN ainda estão em foco, muitos trabalhos estão sendo desenvolvidos e variações nas definições de arquiteturas ainda podem ser encontradas. As pesquisas seguem as mais variadas áreas das redes de computadores, como segurança, gerenciamento e administração. Com o uso de APIs já consolidadas, como o Openflow e Flowvisor pode-se direcionar as pesquisas para criar aplicativos de controle inteligentes que deem suporte a inovação prometida pela SDN.

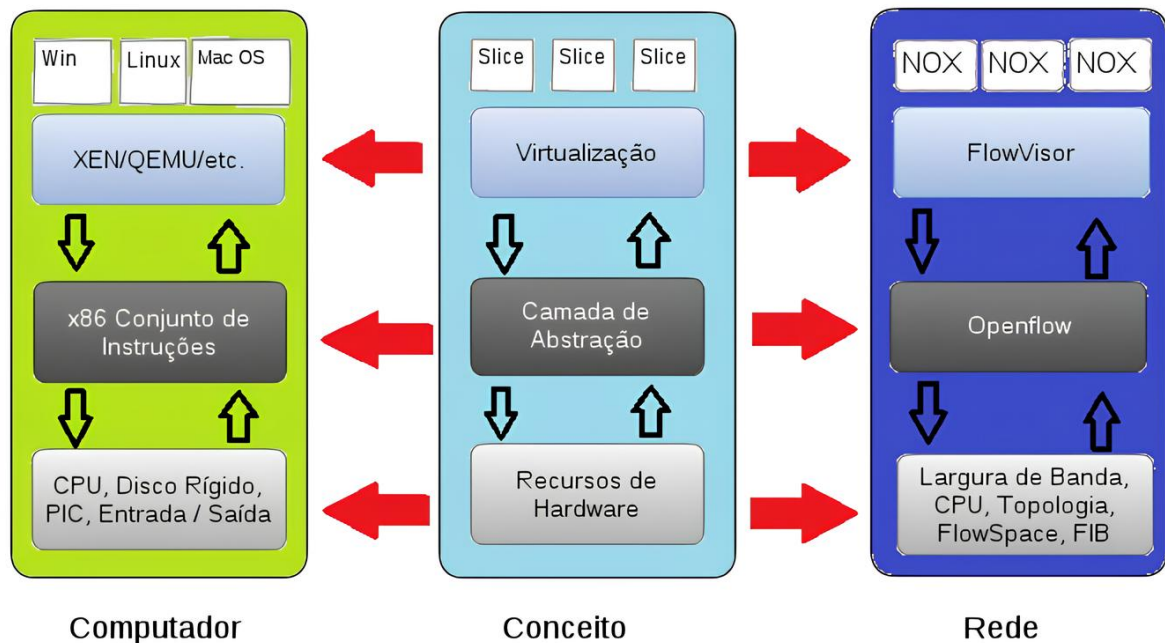
2.2.1 O Flowvisor

A virtualização já é um conceito bastante conhecido para melhoria na alocação de recursos de hardware e segurança em servidores de redes de computadores através do isolamento dos serviços. A ideia de estender o conceito de virtualização também para as redes de computadores é trazer os benefícios da virtualização já conhecidos, somados a possibilidade de fazer implementações em abstrações virtuais de testes antes de aplicar mudanças na rede em funcionamento, permitir o compartilhamento de hardware entre em diferentes redes lógicas, permitir o controle centralizado e isolado de cada instância virtual da rede (PINHEIRO, 2016).

A camada de virtualização deve ser passível de subdivisão, permitindo de diferentes redes virtuais possam compartilhar o servidor de virtualização sem uma causar interferência na outra. A virtualização precisa lidar com essas várias redes virtuais rodando em hardware de rede como comutadores, ponto de conexão sem fio, pontes e outros, de diferentes marcas e modelos, provendo a comunicação delas com os controladores (MENOSSE, 2015).

O Flowvisor é uma API conhecida como solução para o servidor de virtualização, ele é uma camada de abstração que fica entre o comutador de fluxo e o controlador. O Flowvisor deve ser integrado ao Openflow que responsável pela abstração do hardware da rede, para permitir a criação de redes virtuais que possam ser executadas independentes umas das outras (SHERWOOD *et al.*, 2010). A Figura 2.3, proposta em (SHERWOOD *et al.*, 2009) ilustra uma comparação entre a estrutura de virtualização de sistemas operacionais, com o conceito de virtualização de redes e aplicativos usados no processo de virtualização, nota-se que neste exemplo o NOX foi escolhido com software em servidores controladores ligados com o Flowvisor.

Figura 2.3 – Comparação a partir do Conceito de virtualização de rede.



Fonte: (SHERWOOD *et al.*, 2009).

O Flowvisor atua na função de proxy transparente entre os comutadores SDN e os servidores controladores. Todas as mensagens do Openflow (comutador SDN) encaminhadas ao controlador da rede devem passar pelo Flowvisor que decidirá o que fazer seguindo as configurações da rede virtualizada a qual a mensagem pertence, assim os controladores e comutadores SDN entenderam que fazem uma comunicação direta (SHERWOOD *et al.*, 2010).

O Flowvisor é capaz de separar redes em fatias (redes virtuais) com diferentes lógicas de encaminhamento. Isso possibilita que diferentes redes compartilhem os recursos de seus respectivos controladores. Essas redes podem ser definidas com base em diferentes valores dos diversos campos de cabeçalhos que servem para identificar fluxos vindos do hardware através do Openflow. O Flowvisor também pode inserir campos de cabeçalho nos dados do fluxo (GOMES, 2013).

Cada rede virtual deve ser conectada a um controlador SDN, o Flowvisor define cada rede virtual como um conjunto de fluxos de dados chamado de FlowSpace (PINHEIRO, 2016).

2.2.2 O Openflow

O Openflow é um framework, que uma implementação de SDN conhecida, ela vem oferecendo oportunidades para pesquisa na área de testes experimentais de protocolos e softwares de redes para SDN. O Openflow Protocol (OFP), é um protocolo que oferece um controle de tabelas de encaminhamentos de roteadores e comutadores, ele prove uma API

extensível que permite programar o comportamento dos fluxos dos dados das redes, assim possibilitando que os esforços dos pesquisadores se concentrem nos novos protocolos e aplicativos inteligentes para otimizar o controle das SDN ([MCKEWON et al., 2008](#)).

O Openflow é um mecanismo que é executado em todos os comutadores da rede, e permite o isolamento de tráfego. O Openflow executa independente das configurações proprietárias dos equipamentos de rede possibilitando que os fabricantes possam incluir as funcionalidades do Openflow sem necessitarem expor o projeto desses equipamentos, ele possibilita que os comutadores e roteadores possam ser reprogramados sem provocar interferência nas configurações da rede produtiva ([PINHEIRO, 2016](#)).

O Openflow utiliza tabelas com informações de fluxo já existentes nos comutadores convencionais utilizada para implementar serviços como VLANs e NAT. A arquitetura de Openflow pode ser dividida em três partes fundamentais ([PINHEIRO, 2016](#)):

- Protocolo Openflow - OFP: Devido fornecer uma interface externa para atuar sobre os fluxos do comutador esse protocolo permite que o equipamento SDN seja programável e é responsável pela comunicação entre o equipamento SDN e o controlador. O OFP é um protocolo binário aberto.
- Tabela de Fluxos: toda entrada na tabela de fluxo contém em seu cabeçalho uma ação associada, que é utilizada para definir o tipo do fluxo, essas ações definem como um pacote de rede deve ser processado e encaminhados. Nos cabeçalhos podem ser incluídos contadores para gerar informações estatísticas e decidir sobre as remoções de fluxos inativos.
- Canal Seguro: O Secure Channel prove a confiabilidade e confidencialidade na troca de informações entre o controlador e o comutador, ele utiliza o protocolo SSL (Secure Socket Layer) para prover a segurança no tráfego dos dados. Neste canal ainda é possível, de acordo com o controlador, estabelecer interfaces TCP e PCAP (ativas ou passivas), que são robustas e confiáveis para implementações sem a necessidade de chaves de criptografia.

O protocolo Openflow permite que o Software Controlador da Rede, através de um conceito de fluxos de dados, controle os dispositivos físicos de rede compatíveis. Isso dá a possibilidade de segmentar a rede e aplicar a programação nos dispositivos que compõem a rede, isto também dá a possibilidade de serem usados novos segmentos com o propósito de realizar pesquisas e testes que avaliam o desempenho das redes ([RODRIGUES; GUIMARÃES, 2014](#)).

Com separação do plano de controle do plano de dados nos comutadores e roteadores compatíveis com Openflow, permite que rotas sejam reprogramadas sem causar interferência na configuração física da rede. O Openflow deve ser executado em todos os dispositivos

da infraestrutura da rede isolando os tráfegos que podem se distinguir um do outro separando cada tráfego de acordo com a sua funcionalidade definida via software. O Openflow dá a possibilidade de se programar os componentes de rede sem a necessidade de exposição do projeto desses dispositivos, por exemplo, uma nova configuração de Rede pode ser testada em um nó simulado antes de ser implementado como a nova configuração dos enlaces da rede, verificando o impacto de um novo nó antes mesmo dele ser instalado (FARIAS *et al.*, 2018).

Considerando que os switches têm uma parte de hardware e outra de software, o Openflow constitui um canal por onde o controlador pode se comunicar com os switches fazendo alterações em seu software. Através do controle de software o NOS pode ter o acesso e o controle das tabelas de encaminhamento dos switches controlando assim envio de dados.

O Controlador SDN verifica se os pacotes que são encaminhados ao switch pertencem aos fluxos da tabela de encaminhamento daquele switch, quando um pacote pertence a algum dos fluxos ele é enviado de acordo com as regras que já foram pré estabelecidas, se porventura o pacote não pertencer fluxo pré estabelecido para aquele switch ele é enviado de volta ao controlador para que se especifique as regras no pacote. Funções como esta só podem existir através do protocolo Openflow, que deve ter a versão cliente instalada nos switches (SILVA *et al.*, 2013).

O Openflow disponibiliza cabeçalhos destinados a organização de fluxos, através de regras e ações específicas e simples, para encaminhar, descartar, enviar para o controlador ou mesmo reescrever campos do cabeçalho, através das combinações desses cabeçalhos pode se ter as tomadas de decisões, que também são usadas para implementar serviços de NAT, VLANs e firewalls (ROTHENBERG *et al.*, 2010).

O Openflow possui controladores que podem se comunicar com o plano de dados usando mensagens OF, estas mensagens podem ser síncronas e assíncronas. As mensagens síncronas tem início no controlador e exigem resposta ou confirmação. Segundo (SPALLA, 2015) algumas dessas são:

- **Features Request:** Quando uma conexão é estabelecida entre o controlador e o dispositivo de rede o controlador envia uma mensagem de requisição ao dispositivo, que responde enviando dados sobre a sua capacidade e suas configurações.
- **Flow Mod:** Estas mensagens são enviadas pelo controlador e servem para gerenciar os dispositivos de rede, adicionando, modificando, excluindo fluxos nas tabelas dos equipamentos.
- **Stats Request:** Estas requisições recolhem estatísticas, portas e fluxos das tabelas dos dispositivos de rede, que serão utilizadas pelos controladores.

- **Barrier Request:** Utilizadas pelo controlador para garantir o procedimento de pacotes de acordo com uma sequência e para confirmar a realização de determinadas operações.
- **Packet In:** Essa mensagem é enviada ao controlador sempre quando um fluxo não encontra uma entrada na tabela de fluxos para referenciar um pacote de dados. Devido a lógica de encaminhamento está localizada no plano de controle, é o controlador que avalia os pacotes e define o caminho que eles seguirão.
- **Port Status:** Informa ao controlador sobre possíveis mudanças que ocorreram nas portas dos dispositivos de rede como algum cabo que foi desconectado ou alguma desabilitação de interface.
- **Flow Removed:** Informa aos controladores quando uma regra foi removida ou quando irá expirar por inatividade ou por tempo, essas informações sobre as regras são indicadas quando uma regra é criada na tabela de fluxos.

No Framework Openflow existem duas APIs que fazem a interface entre as aplicações, camada de controle e dispositivos. O que facilita o desenvolvimento de aplicações que utilizam controladores SDN com Openflow, são as interfaces Norte e Sul.

2.2.3 As interfaces Norte e Sul

As interfaces de programação de aplicações, (API) são responsáveis por facilitarem o processo de comunicação e interação entre as tecnologias utilizando um padrão de regras, e rotinas, que trocam informações entre as aplicações, favorecendo o desenvolvimento de ferramentas através de informações de seus códigos de fabricantes que podem ser acessadas pelos desenvolvedores através de caminhos fornecidos pelas próprias APIs. (FARES, 2021)

As SDN (Software-Defined Networking) são caracterizadas pela centralização do controle e separação dos planos de encaminhamento de dados. Os controladores executam suas funções via interfaces, que dependem da padronização de APIs. As implementações visam trazer funcionalidades de uma rede virtual independente dos fabricantes dos equipamentos, viabilizando a programação desses dispositivos pelos próprios fabricantes ou administradores. As APIs são classificadas como Northbound e Southbound. (GONDIM, 2015)

Northbound API, trata-se da API norte localizada entre a camada de controle e aplicação, é responsável por fazer a comunicação entre as aplicações e permitindo a troca de informações. (SOUZA, 2019). A interface Northbound é responsável por permitir a comunicação entre os módulos do controlador e as aplicações, além de expor modelos de dados e funcionalidades para o uso de aplicativos. Ela geralmente é de código aberto, mas pode haver exceções em soluções proprietárias, como o Onix. (MOURA, 2015).

Southbound API funciona como uma ponte entre os planos de controle e de dados e um elemento vital para a separação desses planos, por meio desta API os controladores comunicam requisições de aplicações e são transportadas para a rede para fazer a reprogramação de equipamentos para a configuração de suas funções. Dentre essas funções estão, roteamento, comutação, sistemas de detecção de intrusos, firewall e controle de fluxos. A reprogramação é feita com remoção ou adição de regras nas tabelas de fluxos e faz a comunicação dos equipamentos de rede com os controladores. (FARES, 2021)

O Openflow prove a abstração do hardware, ele repassa as informações do fluxo proveniente dos hardware da rede para a camada de Virtualização, o Flowvisor, que recebe os fluxos de dados com as informações necessárias para repassar os devidos fluxos as redes virtuais que serão designadas aos controladores SDN, o esquema simplificado está ilustrado na Figura 2.3.

2.2.4 O Mininet - Arcabouço para implementação e testes práticos em SDN

O Mininet (MININET, 2023) representa um arcabouço de notável relevância, capacitando a criação de ambientes de rede virtual de altíssima fidelidade. Este ambiente é capaz de executar, com primor, os componentes fundamentais de uma rede, tais como o kernel, o switch e o código de aplicativos legítimos. Essa notável capacidade é disponibilizada em uma única máquina de computação, independentemente de se tratar de uma máquina virtual, uma instância de nuvem ou um sistema nativo. A singularidade desse feito reside na sua habilidade de estabelecer essa complexa configuração com uma facilidade inigualável, pois demanda apenas a emissão de um único comando.

A flexibilidade inerente ao Mininet transcende o âmbito técnico e se estende às interfaces de comando e programação de aplicativos, conhecidas como CLI e API, respectivamente. Isso proporciona aos usuários não apenas a capacidade de interagir de forma direta com a rede virtual, mas também a habilidade de ajustá-la para atender a requisitos específicos. Adicionalmente, a natureza colaborativa e facilmente compartilhável do Mininet simplifica a disseminação de conhecimento entre pares, o que culmina em sua utilização diversificada. Esse arcabouço é aplicável em variados contextos, desde o desenvolvimento de software até a instrução educacional e pesquisas de vanguarda (MININET, 2023).

Além disso, o Mininet representa uma plataforma inestimável para a exploração e experimentação de sistemas relacionados a Redes Definidas por Software (SDN), com destaque para as tecnologias OpenFlow e P4. Essa característica singular confere ao Mininet um valor inestimável para aqueles envolvidos no desenvolvimento, compartilhamento e avaliação de projetos voltados à SDN. Para obter acesso a todo seu conteúdo, softwares e manuais basta acessar o site oficial [Mininet.org](https://mininet.org).

Importante notar que o Mininet opera sob os auspícios de uma licença BSD Open

Source permissiva, fomentando, assim, um ambiente de colaboração e contribuição. O projeto em Mininet.org ([MININET, 2023](#)) convida ativamente tanto indivíduos quanto organizações a participarem desse projeto, fornecendo contribuições de código, reportando e corrigindo problemas, melhorando a documentação e adotando quaisquer medidas que visem aprimorar ainda mais a funcionalidade desse sistema singular.

2.3 Inteligencia Artificial e o Aprendizado de Máquina

A Inteligência Artificial (IA) trouxe à tona a ideia de que um dia os computadores possam pensar como um ser humano. Nesta área de estudo apareceu o interesse de automatizar tarefas que demandam algum tipo de raciocínio lógico ([CHOLLET; FRANCOIS, 2018](#)).

[Turing \(1950\)](#) idealizou os principais conceitos impulsionadores das pesquisas de IA deixando vaga a resposta para a ideia de se os computadores seriam um dia capazes de pensar ou mesmo criar regras lógicas apenas analisando um conjunto de dados.

As técnicas de I.A. conhecidas hoje são utilizadas para fazer classificação e categorização de informações, a regressão para fazer estimação de valores a partir de grandes bases de dados e otimizar as variáveis de projeto em função matemáticas. Exemplos de técnicas de I.A são Lógica Fuzzy, Algoritmos Genéticos, Redes Neurais, Redes Bayesianas, Colonias de Vaga-lumes e etc. Já as técnicas de classificação e estimação introduziram a área de aprendizado de máquina.

Segundo [Chollet e Francois \(2018\)](#), a construção de um aprendizado de máquina precisa de 3 grupos de informações sendo os dados de entrada, exemplos de saída e um conjunto de dados para teste de avaliação sobre a qualidade obtida durante o processo de aprendizado.

Os algoritmos de aprendizado de máquina fazem a busca automática de mapeamentos de dados em representações de informações. Destaca-se 3 tipos de aprendizado de máquina, o aprendizado supervisionado, o não supervisionado e o aprendizado por reforço ([BARELLA, 2020](#)).

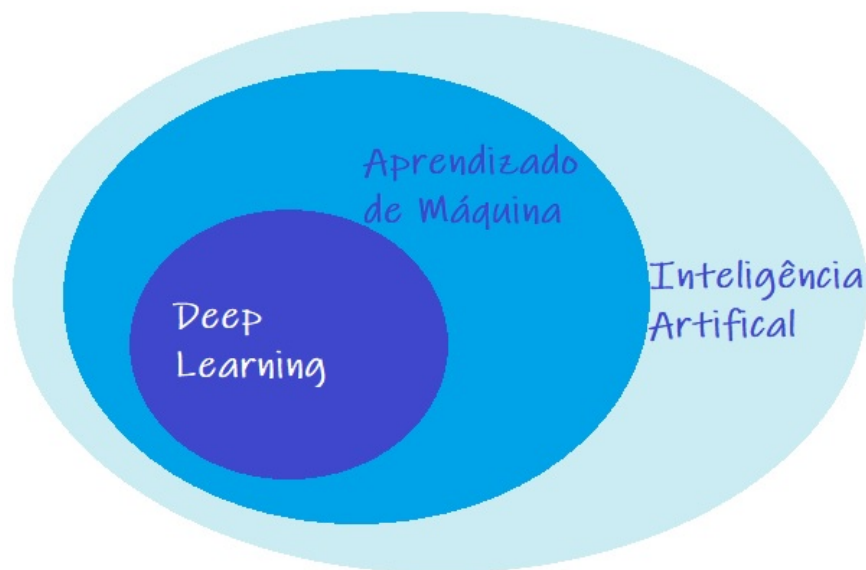
No aprendizado supervisionado exemplos corretos dos padrões a serem aprendidos são fornecidos ao algoritmo, muito utilizado no aprendizado de classificação de dados com por exemplo da categorização de tipos de vinho. Já o aprendizado não supervisionado os dados de entrada não são fornecidos junto de um padrão para aprendizado, o algoritmo precisa fazer a identificação de padrões dentro dos dados, agrupa-los, fazendo a identificação de elementos fora dos padrões presentes, utilizados por exemplo para encontrar danos em estruturas bem definidas ([BARELLA, 2020](#)).

No aprendizado por Reforço o sistema de aprendizagem possui agentes de software que fazem o monitoramento das ações do algoritmo inteligente, esses agentes executam

ações de punição ou recompensa, como o decremento ou incremento de valores em variáveis, uma política de recompensas deve ser definida de acordo com o problema modelado. Um exemplo de uso é o de robôs que precisam aprender a forma de movimentar suas pernas mecânicas para andar minimizando as quedas (GERON, 2019).

Dentre as técnicas de aprendizado de máquina podem ser destacadas as Redes Neurais, essas possuem camadas de processamentos de dados chamadas de neurônios. Técnicas de aprendizagem multicamadas podem ser utilizada para modelar de forma efetiva problemas mais complexos com a mineração de dados em *Bigdata*, toda via esses tipo de modelagem demanda um alto custo computacional. Neste seguimento, o estudo e desenvolvimento de solução com um número muito grande de camadas hierárquicas é denominado de Aprendizado Profundo (Deep Learning), normamento possuem de dezenas a centenas de camadas (CHOLLET; FRANCOIS, 2018). A Imagem 2.4 mostra a relação entre as área da I.A, Aprendizado de Máquina e Aprendizado Profundo.

Figura 2.4 – Relação entre as áreas de pesquisa de I.A, Aprendizado de Máquina e Deep Learnig.

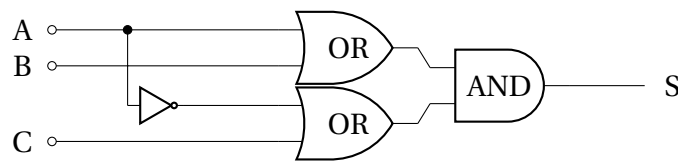


Fonte: (BARELLA, 2020)

Considerando a natureza da solução de Inteligência Artificial voltada para a categorização, estimação e otimização de valores, esta pesquisa adotará a abordagem da Lógica Fuzzy para a inferência de parâmetros de configuração relacionados à largura de banda de rede. Essa inferência será realizada com base em variáveis que estão intrinsecamente relacionadas ao uso em tempo real da rede.

2.4 A Lógica Nebulosa (Lógica Fuzzy)

A lógica Fuzzy é um ramo da lógica que lida com a imprecisão e incerteza presentes em muitos sistemas do mundo real. Ela foi introduzida por Lotfi Zadeh em 1965 como

Figura 2.5 – Sistema Lógico Binário: $(A + B).(\neg A + C)$ 

Fonte: Autor

uma alternativa à lógica binária tradicional (verdadeiro ou falso). A lógica Fuzzy permite representar e manipular informações vagas ou parciais usando graus de pertinência. A lógica Fuzzy tem aplicações em diversas áreas, como controle de sistemas, tomada de decisões, processamento de imagem, análise de dados e muito mais. Muitos estudos têm explorado suas aplicações práticas e suas vantagens em situações onde as abordagens tradicionais de lógica clássica não são adequadas (RIGNEL; CHENCI; LUCAS, 2011).

2.4.1 Lógica binária vs lógica Nebulosa

A Lógica Booleana, concebida por George Boole no século XIX, constitui um sistema de operações que emprega exclusivamente dois valores de entrada: verdadeiro e falso. A influência dos estudos de Boole se estendeu para a consolidação da Lógica Matemática. Durante as investigações na primeira metade do século XX, relacionadas ao desenvolvimento da computação, optou-se pelo sistema numérico binário devido à sua representação simbólica simplificada, que se baseia em apenas dois dígitos: 0 e 1. Este sistema lógico, proposto por Boole, encaixou-se de maneira perfeita, com o valor 0 correspondendo a falso e o valor 1 a verdadeiro. As operações da Lógica Booleana emergiram como um dos fundamentos primordiais da Ciência da Computação. (MARTINS, 2009).

A Lógica Booleana se destaca por suas operações rigorosamente definidas, nas quais as entradas e saídas assumem valores precisos, limitados a 0 ou 1. Essas operações binárias são executadas através de componentes conhecidos como portas lógicas, sendo as principais operações utilizadas no contexto computacional o AND (E), OR (OU), NOT (NÃO) e XOR (OU-EXCLUSIVO). (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

Mediante a combinação de múltiplas entradas e diversas portas lógicas, é possível criar circuitos lógicos que possuem uma capacidade substancial de representação. No entanto, é importante ressaltar que, independentemente da complexidade de um circuito lógico, sua saída final será sempre limitada aos valores de verdadeiro ou falso. (TOCCI; WILMER; MOSS, 2011).

É interessante observar que na figura 2.5, mesmo diante de um sistema lógico binário composto por três variáveis de entrada e várias operações intermediárias, a saída S ainda é limitada a um estado de verdadeiro ou falso. Na figura 2.6, é possível consultar uma tabela que apresenta as principais operações envolvidas no sistema, permitindo uma análise mais

detalhada de como as diferentes combinações de entradas afetam o resultado final. Essas representações gráficas e tabulares são fundamentais para compreender e analisar sistemas lógicos complexos, onde a clareza na definição de estados é essencial para a tomada de decisões e o funcionamento preciso do sistema.

Figura 2.6 – Tabelas de Operações Lógicas Booleanas

X = TER OU NÃO O INGRESSO

Y = POSSUIR OU NÃO AO MENOS 16 ANOS

V = POSSUI

F = NÃO POSSUI

Resultado: Verdadeiro (entrar na festa)

Falso: (não entrar na festa)

and - e - &&		
X	Y	X and Y
V	V	V
V	F	F
F	V	F
F	F	F

or - ou -		
X	Y	X or Y
V	V	V
V	F	V
F	V	V
F	F	F

xor - ^		
X	Y	X xor Y
V	V	F
V	F	V
F	V	V
F	F	F

O operador and exige que todas as assertivas sejam Verdadeira, se ao menos um for falsa a expressão inteira se torna Falsa.

O operador or exige que ao menos 1 elemento das assertivas seja verdadeiro para que a expressão seja verdadeira, se todos os elementos forem falsos, a expressão se torna falsa.

O operador xor é o ou exclusivo, nele todas as assertivas iguais sejam de V ou F torna a expressão falsa, ao menos um precisa ser diferente para a expressão ser verdadeira.

Not - Não - !	
X	!X
V	F
F	V

O operador not, inverte o resultado de uma assertiva. Se for verdadeiro, o not faz ser falso, se for falso o not faz ser verdadeiro.

Fonte: Autor em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

O estudo de caso apresentado na figura 2.6 ilustra a permissão de entrada em uma festa com base em dois critérios principais: a posse do ingresso e a idade mínima necessária. Ao observar a figura, fica evidente como essa situação pode ser representada eficazmente por meio de operações lógicas booleanas. Embora um sistema lógico binário possa produzir apenas dois resultados, 0 ou 1, esses resultados podem assumir diferentes significados conforme o contexto.

Além disso, esses resultados binários podem direcionar ações adicionais, como permitir ou negar a entrada na festa. Isso pode ser ampliado para criar um sistema encadeado, onde a saída de um sistema influencia a entrada em outro, conferindo ao sistema um poder significativo de representação e tomada de decisão.

No entanto, é fundamental destacar que, apesar da complexidade e da interconexão

de sistemas, os resultados ainda se resumem a valores exatos de verdadeiro ou falso, garantindo uma decisão clara e precisa ao final do processo.

O sistema booleano é altamente eficaz em sistemas computacionais convencionais, mas sua capacidade de lidar com problemas é limitada quando a incerteza está presente ou quando é necessário representar um grande número de possíveis saídas diferentes.

A lógica multi-valorada, introduzida por Jan Lukasiewicz em 1920, oferece uma extensão valiosa à lógica booleana, permitindo que valores além de binários, como incerteza e gradações, sejam incorporados. Isso possibilita a criação de sistemas mais flexíveis e capazes de lidar com informações ambíguas. A lógica nebulosa (ou lógica fuzzy) é um exemplo de lógica multi-valorada. Ela é especialmente útil para representar e manipular informações imprecisas ou vagas. (SAMPAIO; OLIVEIRA; IGNACIO, 2007).

Essa abordagem é fundamental em áreas como controle de sistemas, tomada de decisões em condições incertas e processamento de linguagem natural, pois a simplicidade da lógica booleana não é suficiente para lidar com toda a complexidade do mundo real. A Lógica Nebulosa - LN oferece uma estrutura para tratar a incerteza de forma computacionalmente viável, atribuindo graus de pertinência a várias categorias ou estados em vez de respostas estritamente binárias. Isso a torna uma ferramenta poderosa em muitos campos da ciência e da engenharia.

A teoria dos conjuntos nebulosos, originalmente desenvolvida por Lotfi Zadeh, surgiu como uma ferramenta valiosa para lidar com informações que possuem aspectos vagos e imprecisos. Posteriormente, Zadeh avançou ainda mais em sua pesquisa ao desenvolver a teoria das possibilidades, a qual se mostrou mais adequada para tratar de elementos de incerteza em informações, especialmente quando essas informações são transmitidas por seres humanos. (SANDRI; CORREA, 1999).

Um exemplo prático desse conceito é a aplicação de um conjunto nebuloso na teoria das possibilidades para modelar a informação "idade avançada". Com essa abordagem, é possível inferir a probabilidade de uma pessoa ser considerada idosa com base na análise conjunta de outras informações do contexto. A combinação dessas teorias desenvolvidas por Lotfi Zadeh, quando implementada em um sistema lógico, é conhecida como lógica Fuzzy (lógica difusa ou lógica nebulosa).

A lógica Fuzzy oferece uma estrutura mais flexível para lidar com a incerteza e a imprecisão inerentes a muitos problemas do mundo real. Ela permite que as variáveis não sejam apenas verdadeiras ou falsas, mas também expressem graus de verdade em uma escala contínua, tornando-a uma poderosa ferramenta em áreas como controle de sistemas, tomada de decisões e modelagem de sistemas complexos.

A utilização da lógica Fuzzy oferece a capacidade de descrever variáveis de maneira que se assemelham mais ao pensamento humano, tornando-a uma ferramenta valiosa para

a criação rápida de protótipos de sistemas e simplificação da aquisição de conhecimento relacionada ao problema em questão, especialmente quando se lida com entradas que possuem certo grau de incerteza.

Ao contrário da lógica binária tradicional, onde as variáveis são estritamente definidas como verdadeiras ou falsas, a lógica Fuzzy permite que variáveis sejam representadas em termos de graus de pertinência, refletindo a natureza contínua e flexível das informações que muitas vezes é encontrada no mundo real. Essa característica torna a lógica Fuzzy uma técnica bem-sucedida no desenvolvimento de sistemas que controlam processos envolvendo entradas não binárias.

Em sistemas de controle, tomada de decisões e modelagem de sistemas complexos, a lógica Fuzzy se destaca, pois é capaz de lidar de forma eficaz com informações imprecisas, vagas ou ambíguas, tornando-a uma escolha valiosa para cenários em que a lógica binária tradicional não é suficiente para representar adequadamente a complexidade e a incerteza dos dados. (SANDRI; CORREA, 1999) (SAMPAIO; OLIVEIRA; IGNACIO, 2007)

2.4.2 Lógica Fuzzy

A lógica fuzzy tem a notável capacidade de, a partir de entradas subjetivas, gerar saídas numéricas que podem ser utilizadas em processos de tomada de decisão computacionais. Isso se revela extremamente útil em situações em que os conceitos manejados pela humanidade não são sempre objetivos e claros. Em um sistema de recuperação de informações, por exemplo, a tarefa envolve a extração de informações de documentos e a agrupação dos documentos relevantes. No entanto, frequentemente não se dispõe de uma ideia precisa e definitiva sobre a informação necessária, e o próprio conceito de relevância é subjetivo e altamente dependente do contexto.

Nessas situações, a lógica fuzzy permite que os sistemas considerem a ambiguidade e a subjetividade inerentes às decisões humanas. Ela pode atribuir graus de pertinência a diferentes documentos ou informações, levando em conta a incerteza e a subjetividade associadas ao julgamento humano. Isso facilita a classificação e a recuperação de informações de forma mais adaptável e próxima à maneira como os seres humanos lidam com tarefas complexas e ambíguas. Portanto, a lógica fuzzy é uma ferramenta valiosa na área de recuperação de informações e em muitos outros domínios onde a subjetividade desempenha um papel significativo. (FERNEDA; DIAS, 2013).

Na tomada de decisões com informações processadas em modelos subjetivos a lógica Fuzzy tem uma opção de saída que oferece um posicionamento entre um espectro de opções, como, por exemplo, um texto que é 20% irrelevante e 80% relevante, com base nestes dados, pode-se classificar os textos por sua relevância média de acordo com a subjetividade da definição do que é relevante para o usuário, e não apenas em relevante e não relevante

(binário).[\(RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022\)](#)

A Fuzzificação é responsável por converter entradas numéricas ou qualitativas em valores fuzzy, ou seja, em termos de graus de pertinência ou probabilidades. O objetivo da fuzzificação é traduzir informações do mundo real, que podem ser imprecisas ou subjetivas, em uma forma que o sistema Fuzzy possa entender e processar, esse processo será exibido a seguir.

2.4.2.1 Fuzzificação

Quando um usuário introduz uma entrada no sistema, é necessário que essa entrada passe por um procedimento que a converta em uma afirmação de variável linguística, transformando um valor determinístico em um valor difuso. Essa transformação é efetuada por meio da associação da entrada, representada como 'x', a um valor difuso. Para realizar essa conversão, a entrada deve ser submetida a uma função de pertinência. Como resultado desse processo, obtém-se componentes expressos em termos de uma variável linguística. Esse procedimento é conhecido como fuzzificação. [\(ROSS, 2010\)](#).

No contexto da lógica Fuzzy, é fundamental reconhecer que a definição dos termos de entrada como variáveis linguísticas, por meio do processo de fuzzificação, representa um dos primeiros passos essenciais no processamento.

A habilidade de distinguir entre uma variável linguística e uma variável numérica é de suma importância ao empregar a lógica Fuzzy. Isso ocorre porque a fuzzificação permite que informações numéricas sejam interpretadas e tratadas de acordo com a lógica fuzzy, permitindo ao sistema lidar eficazmente com a imprecisão e a subjetividade inerentes a muitos problemas do mundo real. Portanto, a distinção entre variáveis linguísticas e numéricas desempenha um papel essencial na aplicação bem-sucedida da lógica Fuzzy.

2.4.2.2 Variável linguística e variáveis numérica

Em termos computacionais, uma variável se refere a um espaço na memória RAM associado a uma aplicação que armazena um valor. Normalmente, as variáveis são de natureza binária ou numérica, e as variáveis do tipo caractere representam um número inteiro com sua referência na Tabela ASCII.

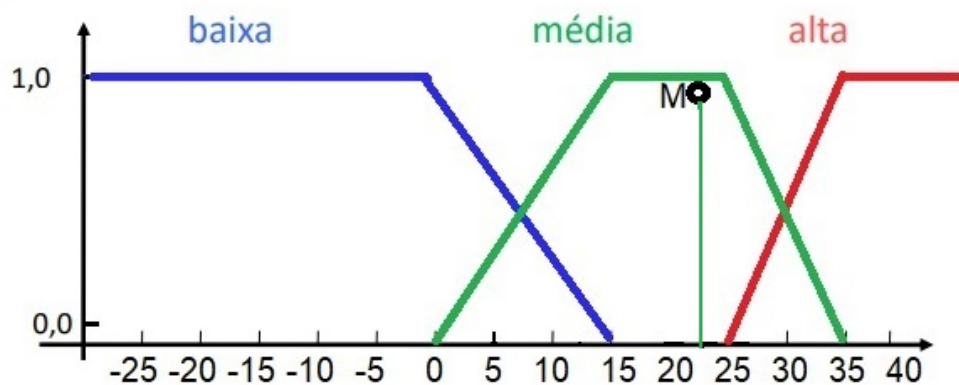
No entanto, diferentemente de uma variável numérica convencional, uma variável linguística, conforme concebida por seu criador Lotfi Zadeh, é caracterizada por conter palavras ou sentenças em linguagem natural ou artificial em vez de valores numéricos. A definição de uma variável linguística pode ser expressa por meio de uma quádrupla $(X, U, T(X), M)$, onde X é o nome da variável, U é o universo de discurso de X , $T(X)$ é o conjunto de nomes para os valores de X e M é o significado dos termos linguísticos quando representados em U através da função de pertinência [\(SANDRI; CORREA, 1999\)](#).

Tendo como exemplo a definição da sensação de calor (informação subjetiva) dado uma temperatura (variável numérica), na figura 2.7 é visto um exemplo de variável difusa (linguística) que dado uma temperatura de 22°C que o seu significado linguístico é que é uma sensação térmica média. O eixo X é o valor de entrada de x e o eixo Y é o valor da função de pertinência para o ponto x .

Figura 2.7 – Variável Nebulosa (linguística)

Variável Linguística (Nebulosa)

- $X = \text{temperatura}$
- $T(X) = \{\text{baixa, média, alta}\}$
- $U = \{x \in X \mid -25 \leq x \leq 40\}$
se $x = 22$:
- $M = \text{Temperatura } 100\% \text{ média}$



Fonte: Autor, adaptado de (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

A definição do ponto $x = 22$ como temperatura 100% média, vem através de uma associação feita pela função de pertinência, nesse caso Trapezoidal, que pode ser observada na figura 2.7, nos seguimentos de linhas azul, verde e vermelho como uma função trapezoidal. (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

2.4.2.3 Função de Pertinência

As funções de pertinência são uma parte fundamental da teoria Fuzzy. Elas descrevem a relação entre um valor de entrada e o grau em que esse valor pertence a um conjunto Fuzzy. As funções de pertinência podem assumir diferentes formas, como Triangular, Trapezoidal, Gaussiana, Singleton, entre outras, dependendo da necessidade encontrada no contexto da aplicação (TANSCHKEIT, 2005).

Na lógica Fuzzy, uma função de pertinência desempenha o papel fundamental de associar um valor " x " pertencente a um universo de discurso " U " a um nome ou termo do

conjunto nebuloso " $T(X)$ ". As curvas ou linhas que compõem a função de pertinência têm como finalidade definir o grau de pertinência, ou seja, estabelecem a associação de um valor numérico no intervalo de 0 a 1. Esse valor representa a proporção ou grau pelo qual o valor " x " pertence ao conjunto nebuloso " $T(X)$ ". Onde 0 significa que não pertence a um termo $T(X)$, 1 pertence completamente e um valor entre 0 e 1 pertence parcialmente com diferentes níveis de intensidade, quanto mais perto de 1 melhor se encaixa ao termo $T(X)$ em questão (SANDRI; CORREA, 1999).

Essas funções de pertinência são essenciais para a modelagem de variáveis linguísticas em sistemas Fuzzy, permitindo a representação de incerteza e a quantificação de quão bem um valor " x " se encaixa em um determinado termo linguístico. Esse grau de pertinência é fundamental para a tomada de decisões e o processamento de informações em sistemas Fuzzy, onde a ambiguidade e a imprecisão são levadas em consideração de forma mais adequada do que na lógica binária convencional.

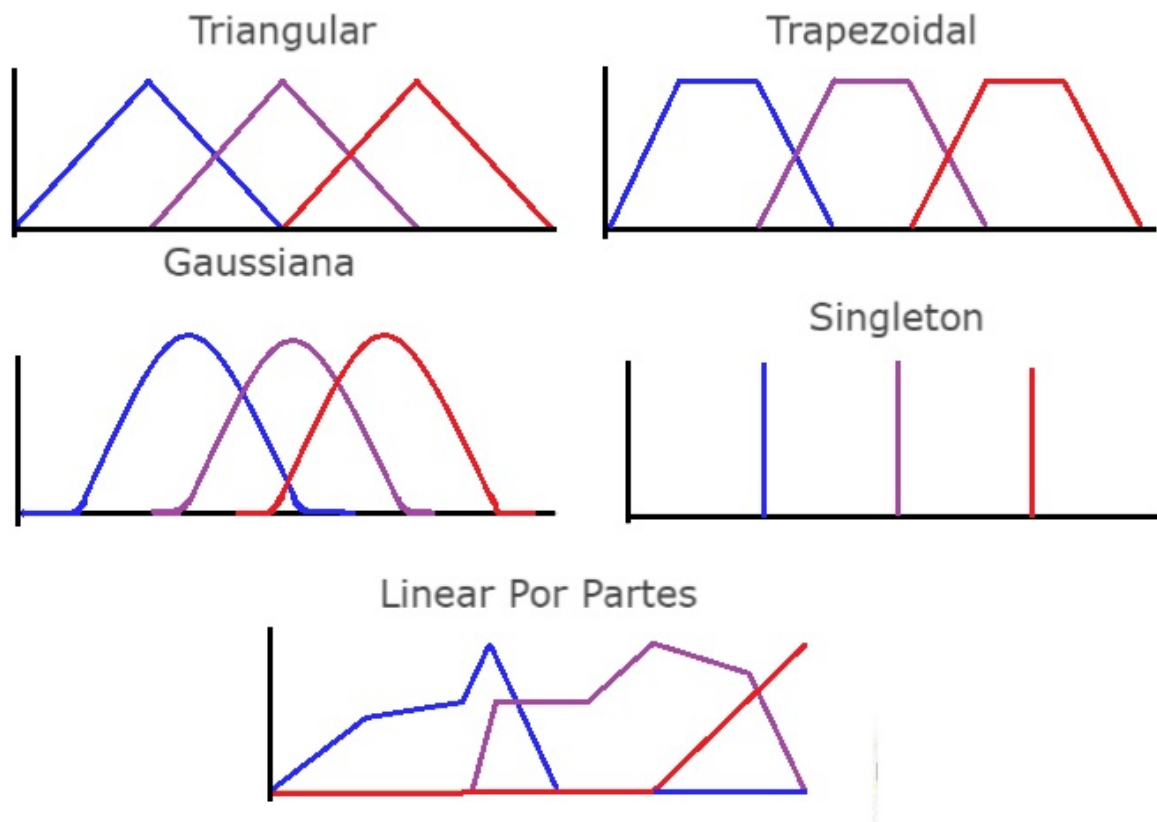
Ao utilizar a função de pertinência da figura 3.1 como exemplo, se $x = 10$, M terá seu grau de pertinência em aproximadamente 30% temperatura fria e 70% temperatura média. (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

Existem vários tipos de funções de pertinência, e a escolha da função apropriada depende inteiramente do problema específico que está sendo modelado. Cada função de pertinência pode se adequar melhor a determinadas situações ou tipos de variáveis linguísticas. As funções de pertinência mais conhecidas são: trapezoidal, triangular, gaussiana, singleton e linear por partes (YILDIZ, 2010). A figura 2.8 mostra cada uma delas.

É possível verificar na figura 2.8 que cada tipo de função de pertinência possui características distintas que afetam a forma como os valores de pertinência são distribuídos ao longo do universo de discurso. Pode-se destacar sobre cada uma:

- **Função Trapezoidal:** Permite que alguns valores de " x " atinjam 100% de pertinência em um intervalo, ao mesmo tempo em que mantém uma transição gradual e mistura de graus de pertinência na interseção. Isso é útil quando a pertinência plena a um conjunto fuzzy é válida em um intervalo contínuo.
- **Função Triangular:** Apresenta uma transição rápida entre os valores de pertinência, resultando em uma faixa muito estreita em que o grau de pertinência é 100%. Essa característica é adequada quando se deseja uma pertinência plena em um intervalo muito específico.
- **Função Gaussiana:** Permite que os valores de pertinência se misturem suavemente, tornando improvável que qualquer valor atinja 100% de pertinência. Isso reflete a natureza contínua e suave da função Gaussiana, adequada para situações em que a pertinência é gradual e suave.

Figura 2.8 – Tipos de Função de Pertinência



Fonte: (YILDIZ, 2010) (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

- **Função Singleton:** Cada elemento difuso é 100% pertinente em um único ponto, sem mistura. Isso é útil quando se deseja representar pertinências absolutas e discretas.
- **Função Linear por Partes:** Oferece flexibilidade para personalizar a pertinência ao longo do universo de discurso, com segmentos lineares. Isso permite modelar diferentes comportamentos de pertinência de acordo com as necessidades do problema. Possuindo um padrão de comportamento independente entre os elementos de $T(X)$.

A escolha da função de pertinência é vital na modelagem de sistemas Fuzzy, influenciando como os valores de pertinência são distribuídos e como a incerteza é tratada em diferentes contextos. Cada tipo de função de pertinência tem seu lugar e utilidade, dependendo das características do problema que está sendo modelado.

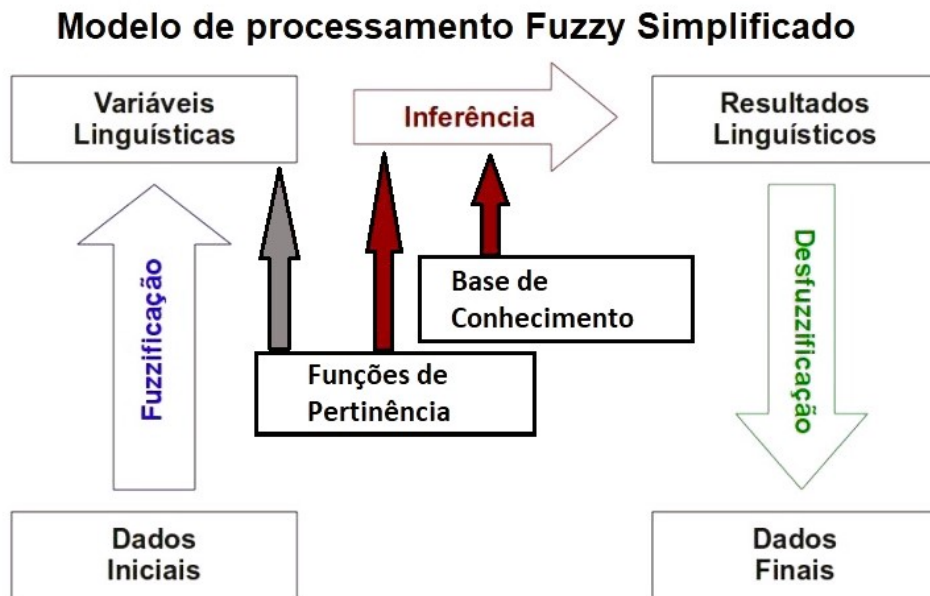
Após a fase de fuzzificação, os dados difusos estão prontos para serem processados pelo sistema Fuzzy, marcando o início da etapa de preparação para a inferência.

Após a conclusão da etapa de fuzzificação, os conjuntos difusos precisam ser repassados à base de regras. Essa submissão é essencial para preparar as variáveis de saída difusas

para o subsequente processo de inferência.

O modelo simplificado do sistema Fuzzy pode ser compreendido conforme ilustrado na Figura 2.9.

Figura 2.9 – Modelo Simplificado de um Sistema Difuso



Adaptado da Fonte: (AGUADO; CANTANHEDE, 2010) (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

2.4.2.4 Base de Conhecimento

A base de regras, ou conhecimento, em um sistema Fuzzy define regras do tipo "se-então-senão". Por exemplo, uma regra pode ser formulada da seguinte maneira: "Se a prioridade do serviço de rede é baixa, então a banda máxima disponível é baixa; senão, a banda máxima disponível é alta." Essas regras estabelecem as relações entre as variáveis de entrada e saída do sistema Fuzzy, permitindo que o sistema tome decisões baseadas na lógica fuzzy e nas informações fornecidas pelos especialistas no domínio do problema.

Essas regras são necessárias para a operação do sistema Fuzzy, pois orientam o processo de inferência, ajudando a determinar os graus de pertinência das variáveis de saída com base nos graus de pertinência das variáveis de entrada e nas relações definidas pelas regras estabelecidas. Isso permite que o sistema Fuzzy tome decisões ou produza saídas difusas que reflitam a complexidade e a incerteza inerentes a muitos problemas do mundo real.

A base de regras difusas é composta por operações realizadas sobre conjuntos difusos. Essas operações envolvem os operadores AND, OR e NOT. As principais variações dessas operações podem ser visualizadas nas Tabelas 2.1, 2.2 e 2.3.

Tabela 2.1 – Operador sob conjunto difuso OR

Máximo	$Max\{\mu_A(x), \mu_B(x)\}$
Soma Probabilística	$\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$
T-conorma de Lukasiesicz	$Min\{1, \mu_A(x) + \mu_B(x)\}$

Fontes: (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003) (YILDIZ, 2010).

Tabela 2.2 – Operador sob conjunto difuso E

Mínimo	$Min\{\mu_A(x), \mu_B(x)\}$
Produto	$\mu_A(x)\mu_B(x)$
T-conorma de Lukasiesicz	$Max\{0, \mu_A(x) + \mu_B(x) - 1\}$

Fontes: (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003) (YILDIZ, 2010).

Tabela 2.3 – Operador sob conjunto difuso NOT

Negação usual	$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$
Negação de Sugeno	$\mu_{\bar{A}}(x) = \frac{1-x}{1+\lambda x}, \lambda \in (-1, +\infty)$
Negação de Yager	$\mu_{\bar{A}}(x) = \sqrt[w]{1-x^w}, w \in (0, +\infty)$

Fontes: (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003) (YILDIZ, 2010).

2.4.2.5 Inferência

Após a aplicação da base de regras às variáveis de entrada, as variáveis de saída são preparadas para o processo de inferência. Os principais métodos de inferência na lógica Fuzzy são o método de Mamdani e o método de Takagi-Sugeno-Kang.

O modelo concebido por Ebrahim Mamdani na década de 70 estabeleceu um padrão significativo na área de sistemas nebulosos. Este modelo é caracterizado por suas regras baseadas na linguagem de processamento natural, tornando a modelagem acessível e possibilitando o processamento de sistemas com variáveis de grau de incerteza. A regra semântica comumente adotada no contexto do banco de conhecimento Mamdani é conhecida como *Máx-Min*, a qual faz uso de operações de interseção e união entre conjuntos difusos (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003). O Código 2.1 apresenta uma descrição do método de inferência Máx-Min.

Código 2.1 – Inferencia Max-Min

```

1 if (x = Ai e y= Aj e ... e xp = Aj):
2 then w = Bi e z = Bm

```

Oliveira, Almeida e Evsukoff (2003) descrevem a função acima com x e y sendo as entradas do sistema, $A_1, \dots, A_j$, com os termos primários definidos nas partições Fuzzy de

cada variável de entrada, w e z como variáveis de saída e $B_1, \dots, B_m$ os termos primários definidos em suas partições Fuzzy.

O modelo Takagi-Sugeno-Kang, originalmente definido em 1985 e comumente abreviado como TSK, atua como um aproximador de sistemas capaz de representar de forma satisfatória as relações entre entradas e saídas. O banco de conhecimento associado ao modelo TSK segue uma sintaxe que implica modificações nas propriedades semânticas, baseadas em um conjunto de regras condicionais de inferência. O Código 2.2 exemplifica a tipificação das regras TSK, conforme definido por (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003).

Código 2.2 – Tipificação das Regras TSK

```

1  if (x1 = Ai e x2 = Aj e ... e xp = Am)
2  then y = f (x1, x2, ..., xp)

```

Tanto o modelo Mamdani quanto o modelo TSK compartilham a etapa inicial de processamento das regras de entrada dos conjuntos Fuzzy. No entanto, a partir desse ponto, seus processos divergem significativamente em suas abordagens e características. (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

O Mamdani é notável por sua descrição próxima à linguagem de processamento natural, resultando em saídas subjetivas que se assemelham às entradas em termos linguísticos. Ele é especialmente eficaz na captura de situações que envolvem a compreensão humana, tornando-se uma escolha adequada para sistemas que precisam refletir o raciocínio humano. (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003) (YILDIZ, 2010)

Por outro lado, o TSK é formulado em linguagem matemática e suas saídas são determinadas por funções matemáticas claras e definidas. Ele é altamente adequado para criar controladores lineares eficazes em modelos cujas funções matemáticas $f(x)$ são conhecidas e podem ser representadas de forma precisa. Portanto, o TSK é uma escolha preferida quando se trata de sistemas onde a modelagem matemática é mais apropriada e as saídas precisam ser calculadas de maneira precisa. (OLIVEIRA; ALMEIDA; EVSUKOFF, 2003) (YILDIZ, 2010)

Ambos os modelos têm suas aplicações específicas e são escolhidos com base nas características e requisitos do problema em questão. O Mamdani destaca-se pela sua interpretabilidade e adaptabilidade à incerteza, enquanto o TSK é preferido quando se busca uma abordagem matemática mais precisa e determinística.

Ao compreender as principais técnicas de inferência, pode-se afirmar que, após o processamento das variáveis de entrada por meio das regras contidas na base de conhecimento, a variável de saída está pronta para ser submetida ao processo de inferência. A inferência envolve a combinação das informações derivadas das variáveis de saída por meio de técnicas de acumulação. Essas técnicas são detalhadas na Tabela 2.4.

Tabela 2.4 – Métodos de Acumulação

Operação	Fórmula
Máximo	$Max\{\mu_A(x), \mu_B(x)\}$
Soma Limitada	$Min\{1, \mu_A(x) + \mu_B(x)\}$
Soma Normalizada	$\frac{\mu_A(x) + \mu_B(x)}{Max\{1, \mu_A(x) + \mu_B(x)\}}$

Fonte: (YILDIZ, 2010)

Após o processo de inferência, a variável de saída está pronta para passar pelo processo de defuzzificação, que a transformará de uma variável difusa em uma variável numérica. Esse passo é imperativo para obter uma resposta clara e quantificável a partir dos resultados difusos gerados pelo sistema Fuzzy.

2.4.2.6 Defuzzificação

Segundo a explicação de (COX, 1994), a desfuzzificação representa a etapa conclusiva do sistema Fuzzy, onde os valores Fuzzy (nebulosos ou difusos) são restaurados para valores nítidos (numéricos determinísticos). Diversos métodos de desfuzzificação estão disponíveis, e eles desempenham um papel fundamental nesse processo:

- Centroid: o método é uma técnica de desfuzzificação na qual a saída é determinada como sendo o centro da mediana do conjunto Fuzzy. (AGUADO; CANTANHEDE, 2010).
- Altura Máxima: também conhecido como "Maximum Height," é uma técnica de desfuzzificação na qual a saída é obtida calculando a média entre os dois elementos extremos no universo de discurso da variável linguística de saída. Esses elementos correspondem aos maiores valores da função de pertinência da variável de saída. (AGUADO; CANTANHEDE, 2010).

Yildiz (2010), define as principais funções de desfuzzificação matematicamente em suas variações, como indicado na Tabela 2.5.

Tabela 2.5 – Principais Operações de defuzzificação

Operação	Fórmula
Centro de Gravidade	$U = \frac{\int_{min}^{max} u\mu(u)du}{\int_{min}^{max} \mu(u)du}$
Centro de Gravidade para <i>Singletons</i>	$\frac{\sum_{i=1}^p [u_i\mu_i]}{\sum_{i=1}^p [\mu_i]}$
Máximo Mais à Esquerda	$U = inf(u'), \mu(u') = sup(\mu(u))$
Máxima Mais à Direita	$U = sup(u'), \mu(u') = sup(\mu(u))$

Fonte: (YILDIZ, 2010)

Os trabalhos relacionados a seguir proporcionam um contexto sobre as potenciais implementações para soluções de Redes Definidas por Software (SDN) já estabelecidas e uma revisão de trabalhos sobre o uso de funções de pertinência.

Capítulo 3

Trabalhos Relacionados

Para verificar as principais abordagens existentes hoje na academia em relação a SDN e as aplicações já existentes neste seguimento foi realizada uma revisão do tema SDN e Lógica Fuzzy nas bases indexadas na forma de revisão da literatura.

Durante a construção da revisão bibliográfica é possível identificar as melhores oportunidades para novas pesquisas. Utilizando as informações encontradas pelos autores da área foi possível direcionar a pesquisa delineando SDN e Lógica Fuzzy como foco neste trabalho.

As principais bases de trabalhos utilizadas nesta revisão foram: CAPES (2023); *Scopus* (2022), *Science Direct* (2022), *ACM* (2022) e *Google Acadêmico* (2023), além de outras bases com resultados parecidos que não precisam ser citados para mostrar abrangência buscada aos termos pesquisados.

A revisão dos trabalhos relacionados concentrou-se em várias áreas-chave. Primeiramente, foi explorada a aplicação das funções de pertinência na Lógica Fuzzy, destacando sua relevância e contribuições. Em segundo lugar, analisamos estudos relacionados ao gerenciamento e roteamento de redes de computadores, com foco especial em seus protocolos quando aplicados à SDN. Por fim, investigamos a utilização prática da linguagem de programação Python em conjunto com a plataforma Mininet e a Lógica Fuzzy. Essa análise abrangente dos trabalhos relacionados estabelece uma base para o desenvolvimento e a contribuição deste trabalho, identificando oportunidades de inovação no campo das Redes Definidas por Software e Lógica Fuzzy.

3.1 Aplicação de funções de Pertinência na Lógica Fuzzy

A literatura tem explorado a definição, propriedades e aplicação de funções de pertinência em sistemas Fuzzy. Muitas pesquisas estão focadas em melhorar a modelagem de funções de pertinência para que elas capturem com precisão a incerteza presente nos dados

do mundo real. A lógica Fuzzy tem sido amplamente utilizada no controle de sistemas complexos, como sistemas de automação, robótica e processos industriais. Sistemas de controle Fuzzy podem lidar bem com variações e incertezas presentes nesses sistemas.

A lógica Fuzzy é aplicada na tomada de decisões em ambientes complexos e incertos. Ela permite que informações imprecisas sejam consideradas de maneira mais realista. A lógica Fuzzy também é usada para lidar com a ambiguidade e incerteza presentes em tarefas de processamento de linguagem natural, como a análise de sentimentos em textos. Em sistemas de recomendação, a lógica Fuzzy pode ser usada para lidar com preferências vagas e incertas dos usuários.

[Hasan e Sobhan \(2020\)](#) desenvolveram um método inovador para criar funções de pertinência Fuzzy a partir de um conjunto de dados, utilizando um resumo de cinco números. Esse método permite a criação de funções de adesão de cinco estados, ao mesmo tempo em que oferece a capacidade de identificar valores discrepantes no conjunto de dados. Geralmente, o box plot é empregado para detectar tais valores discrepantes. Nesse contexto, o box plot foi utilizado em conjunto com a nova abordagem, corroborando a precisão dos resultados alcançados pelo novo método. Para validar a eficácia do método proposto, diversos exemplos do cotidiano foram analisados, acompanhados de gráficos que destacam o potencial da abordagem.

Os resultados obtidos demonstraram o bom desempenho do método. Mesmo em situações com valores discrepantes, o método proposto e o *box plot* apresentaram resultados comparáveis. Uma das principais vantagens desse procedimento inovador é sua eficiência econômica em relação a alternativas como redes neurais e algoritmos genéticos. A pesquisa de [Hasan e Sobhan \(2020\)](#) introduz um método promissor para construção de funções de pertinência Fuzzy a partir de resumos de cinco números. A abordagem é validada através de exemplos práticos e comparações com o método do *box plot*, demonstrando sua robustez e utilidade.

[Arslan e Kaya \(2001\)](#) contribuíram ao ajustar as funções de pertinência para uma nova perspectiva do estudo, visando determinar os tipos ideais dessas funções. Eles empregaram um Algoritmo Genético, uma técnica de busca adaptativa fundamentada no conceito de seleção natural de regras genéticas. Os Algoritmos Genéticos encontram aplicação em várias áreas científicas e podem, igualmente, ser aplicados na determinação das funções de pertinência em sistemas Fuzzy.

[Prajapati e Fernandez \(2020\)](#) realizaram uma avaliação comparativa de distintas funções de pertinência difusas, visando analisar o desempenho de saída de um painel solar fotovoltaico. O estudo contemplou a aplicação de três funções de associação - triangular, trapezoidal e gaussiana - para fuzzificar os dados relacionados ao rendimento do painel solar em três níveis de temperatura distintos.

A avaliação de desempenho foi realizada com base na potência gerada pelo painel, abrangendo níveis de radiação solar que variam entre 200 W/m² e 1000 W/m². A seleção da função de pertinência mais adequada para o modelo foi determinada por meio do cálculo do erro quadrático médio, em comparação com os resultados do banco de dados de referência. Um dos desafios fundamentais da etapa de fuzzificação dos dados de entrada consiste na escolha de uma função de pertinência apropriada. A forma da função de pertinência exerce influência direta sobre o comportamento do sistema de inferência difusa específico (PRAJAPATI; FERNANDEZ, 2020).

No estudo de Pelalak *et al.* (2021), foi conduzido um projeto numérico de um reator de coluna de bolhas, incorporando um único aspersor posicionado no centro do reator. Após essa etapa, determinados parâmetros de entrada e saída foram selecionados durante o pós-processamento da abordagem numérica. Em seguida, a análise de aprendizado de máquina foi empregada para examinar o grau de complexidade e o impacto de cada parâmetro de entrada sobre os resultados de saída. Para abordar o fluxo gás-líquido no reator, a equipe explorou o método adaptativo de inferência neuro-Fuzzy como uma abordagem de aprendizado de máquina. A técnica ANFIS (Adaptive Neuro-Fuzzy Inference System) foi utilizada para simular o fluxo tridimensional em um reator de coluna de bolhas. Esse modelo foi empregado para analisar conjuntamente a interação dos parâmetros de entrada e saída.

Especificamente, o estudo examinou como o grau das funções de pertinência varia em relação a cada entrada, investigando a complexidade da fração de gás com base nos nós de computação nas direções X, Y e Z. Os resultados indicaram que um maior número de funções de pertinência está associado a uma compreensão mais aprofundada do processo, resultando em maior precisão e capacidade de previsão do modelo. No que tange à influência nos resultados, os nós de computação X e Y demonstraram impacto similar na fração de gás, enquanto os nós de computação Z (correspondentes à altura do reator) apresentaram uma distribuição uniforme das funções de pertinência ao longo da coluna (PELALAK *et al.*, 2021).

Bhattacharyya e Mukherjee (2020) empreenderam uma análise de trinta e três distintas metodologias para avaliar funções de pertinência, abordando publicações no período de 1971 a 2016. Visando resolver essa questão, o presente artigo apresenta um novo algoritmo, fundamentado em um esquema de avaliação de funções de pertinência difusas de caráter não linear, empregando técnicas de análise de regressão e álgebra. Para comprovar a aplicabilidade e a viabilidade do método, foram conduzidos três estudos de caso distintos. Além disso, uma análise comparativa foi realizada em relação à literatura existente, a fim de contextualizar o método proposto e avaliar sua eficácia.

Fayaz, Ullah e Kim (2019) apresentaram uma metodologia inovadora denominada "aprendizagem para controle" para abordar os desafios inerentes ao método de inferência difusa. A estrutura dessa metodologia consistiu em dois módulos principais: um módulo

de algoritmo de controle e um módulo de algoritmo de aprendizagem. No primeiro módulo, empregou-se o método Mamdani, enquanto o segundo fez uso de um algoritmo de rede neural artificial (RNA). As entradas da RNA basearam-se na diferença de erro entre a temperatura ambiente e a temperatura desejada, com a saída da RNA configurada para os conjuntos Fuzzy do controlador (FLC). As entradas do controlador lógico difuso (FLC) foram a diferença de erro entre a temperatura ambiente e a temperatura desejada (D), enquanto a saída correspondia à potência requerida para o atuador do ventilador.

A principal missão da RNA era ajustar os conjuntos Fuzzy do FLC para aprimorar sua eficácia. O método de aprendizagem para controle proposto foi então aplicado aos dados e comparado com o método convencional do controlador de lógica Fuzzy, a fim de avaliar o desempenho do modelo. Os resultados evidenciam que a abordagem proposta supera consideravelmente as técnicas convencionais de lógica Fuzzy em termos de desempenho.

Vaidya, Metkewar e Naik (2018) aprofundaram o desenvolvimento de um algoritmo baseado em lógica Fuzzy, incorporando a funcionalidade de geração automática de funções de pertinência ao módulo de lógica Fuzzy do algoritmo. Nesse âmbito, uma breve revisão de trabalhos correlatos na área de geração de funções de associação é apresentada, e as regras associadas a essas funções são integradas ao contexto. Foi introduzido um método de descoberta de adequação, que foi aplicado para avaliar o algoritmo proposto na derivação de funções de pertinência. Este método baseou-se na utilização de parâmetros de controle durante a implementação. A aplicação do algoritmo foi conduzida considerando um estudo de caso que envolve dados do mercado de ações. Os resultados obtidos foram minuciosamente analisados e comparados com cenários de abordagens mais convencionais.

3.2 Controle SDN e suas aplicações em Redes de Computadores

Segundo Levin *et al.* (2012), SDN proporciona a possibilidade para refatorar o plano de controle da rede, possibilitando que a lógica de controle seja feita utilizando a visão de rede global. Mesmo com a abstração da Rede em uma rede global com redes virtuais individuais e independentes é interessante que o plano de controle deve ser distribuído fisicamente para melhorar a confiabilidade e disponibilidade da rede, melhorando o tempo de resposta provindos do meio físico.

Levin *et al.* (2012) caracterizou os principais pontos de distribuição de estado no plano de controle SDN distribuído encontrando trade-offs entre desatualização e otimização, aplicação e complexidade lógica e robustez e inconsistência. Foi analisado como o estado SDN distribuído fisicamente afeta o desempenho de um aplicativo de controle centralizado no contexto lógico através de simulação e análise dos pontos de trade-offs. Com os resultados obtidos ficou constatado que a desatualização a respeito do estado da rede afeta a otimi-

zação do aplicativo de controle e que quando o aplicativo está consciente da distribuição do hardware a robustez perde força aumentando a inconsistência de software. Fazer a camada de fluxo e seus detalhes de implementação de hardware serem transparentes para os softwares de controle é desejável no contexto SDN, o Openflow cumpre essa função.

[ARBIZA \(2016\)](#) mostra a viabilidade de utilizar SDN no contexto de IoT aplicada no monitoramento de pacientes. Os protocolos e serviços provenientes do Openflow foi a base da conexão da camada de fluxo de dados ao ponto de gerência da rede. Neste cenário o middleware foi refatorado para propiciar um sistema de IoT que utiliza SDN fazendo com que a maior parte do fluxo de serviços e do monitoramento da rede especificada fiquem em servidores dedicados remotos, possibilitando fazer uso de recursos que vão além do seguimento do roteador possibilitando um trabalho de desenvolvimento simultâneo e mais ágil com mais funcionalidades. Adicionando SDN neste cenário foi possível obter uma entrega de múltiplos serviços através de um único ponto de acesso, mais escalabilidade no crescimento de demandas e mais autonomia no gerenciamento da implantação dos serviços, dispositivos e redes.

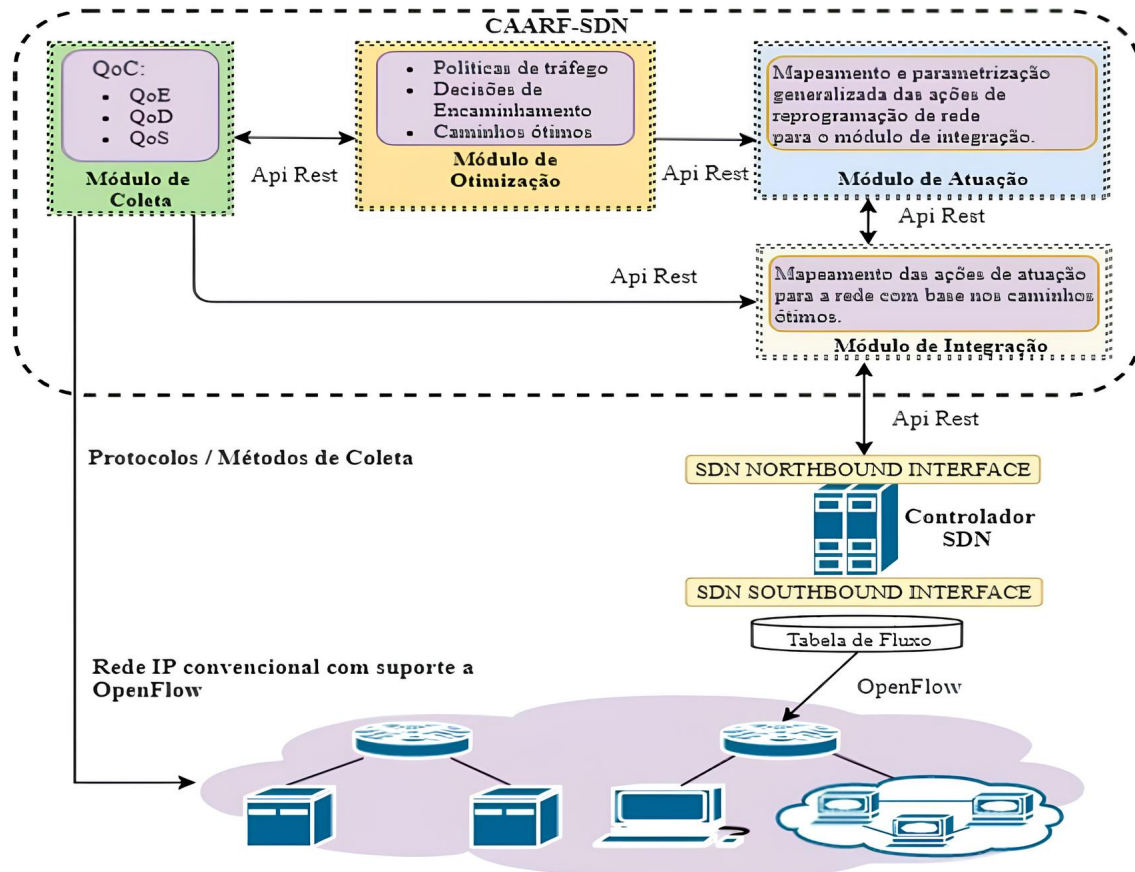
A escolha do controlador SDN interfere diretamente no trabalho do desenvolvimento de software para a gerência das SDNs, sendo que cada um exige a implementação de soluções em linguagens específicas. [Centeno et al. \(2021\)](#) fazem um trabalho comparativos entre diferentes tipos de controladores SDN. Dentre diferentes características exploradas destaco que os controladores POX e Ryu utilizam a linguagem Python e Beacon, Floodlight e ODL utilizam a linguagem Java, NOX utiliza a linguagem C++ e o Trema utiliza a linguagem Ruby/C.

Análogo as tarefas de um Gerente de Redes de Computadores tem-se o Administrador de Segurança em Redes de Computadores, trazendo para o contexto da SDN [Júnior, Tavares e Nogueira \(2019\)](#) mostram o software LPAProg, que faz uso da Lógica Paraconsistente Anotada (LPA) para detecção de ataques de Negação de Serviços Distribuídos (DDoS) do tipo SYN-Flooding em SDNs. Ataques do tipo de negação de serviço (DoS) e suas variantes são ataques recorrentes no cotidiano do gerente de rede, existem diferentes tipos de aplicações de redes capazes de identificar e mitigar um ataque DDoS, toda via a utilização de estratégias para SDN são interessantes. Esse tipo de ataque é mais efetivo servidores que são o centro de diferentes segmentos de redes, que é o caso de um controlador SDN. O LPAProg conseguiu fazer boas previsões a cerca da identificação desses tipos de ataques, restando como um trabalho futuro implementar métodos automáticos de mitigação ([Júnior; TAVARES; NOGUEIRA, 2019](#)).

[Neto, Miguel e Sampaio \(2021\)](#) apresentam o arcabouço Context-Aware Adaptive Routing Framework SDN (CAARF-SDN), que é uma solução de otimização dinâmica de SDN que visa otimizar o tráfego de SDN. Aplicando o CAARF-SDN para otimização de tráfego de VOIP (Voz sobre IP) ele foi capaz de mudar a prioridade de fluxo e topologia baseado nas métricas da QoS, QoE (Qualidade de Experiência) e QoD (Qualidade de Dispositivo). O experi-

mento foi apresentado em um modelo simulado utilizando Mininet. O CAARF-SDN possui 4 módulos principais sendo o Módulo de Coleta, Módulo de Otimização, Módulo de Atuação e Módulo de Integração. A dinâmica do módulos do CAARF-SDN com o Openflow e o Software Controlador SDN pode ser vista na Figura 3.1.

Figura 3.1 – Arquitetura CAARF-SDN.



Fonte: Adaptado de (NETO; MIGUEL; SAMPAIO, 2021)

Para GUTZ *et al.* (2012)) manter o tráfegos de diferentes tipos de protocolos isolados é importante para implementar a QoS (Qualidade de Serviço) e fazer isso não é trivial, normalmente é feito através da programação em hardware ou virtualização como em VLANs. Como alternativa foi apresentada uma implementação SDN para manter diferentes tipos de tráfegos isolados. Utilizando o Openflow para gerenciar a camada de abstração do hardware e a linguagem Python com a biblioteca nxdtopo foi definida e testada uma segmentação de rede com fatias isoladas.

3.3 Roteamento SDN e QoS

O Roteamento, se tratando de uma função de camada de Rede/Internet do modelo TCP/IP em redes tradicionais é feito em equipamento designados como roteadores que através de seu software determina a interface em que os dados devem seguir. Nas SDN essa função, como toda função de software, é feita pelo Controlador SDN e repassada para os comutadores SDN utilizando o Openflow. Neste ponto é justo discutir a respeito da QoS (Qualidade de Serviço) presente na rede, que deve controlar os enlaces distribuindo os dados de maneira que evite congestionamento e provenha um funcionamento da rede de velocidade constante e controlada na transmissão de dados.

De acordo com [Assis e Errico \(2014\)](#), o processo de roteamento em SDN é um dos grandes desafios desse paradigma, pois é o controlador SDN quem escolhe a melhor rota para os fluxos percorrerem até chegarem ao destino, garantindo qualidade nos serviços, balanceamento de cargas e utilização eficiente dos recursos. A comunidade científica tem realizado amplas pesquisas acerca de pontos importantes como performance, balanceamento de carga, soluções que aumentam a largura de banda e diminuem a latência nas redes, tornando mais eficiente o uso dos recursos.

A pesquisa realizada por [Assis e Errico \(2014\)](#) se concentrou na proposta de um algoritmo de otimização multiobjetivo (que explora como variáveis devem atender diferentes funções matemáticas simultaneamente) para roteamento em SDN, com o objetivo de alcançar uma melhor performance e utilização eficiente dos recursos. Através desse algoritmo, foi possível atender aos pontos observados pelas pesquisas científicas, tais como a qualidade dos serviços, balanceamento de carga, aumento da largura de banda, diminuição da latência e melhor eficiência energética.

O algoritmo de otimização estudado tinha como objetivo principal encontrar, em uma única execução, um conjunto Pareto de soluções que considerassem múltiplos objetivos. A abordagem utilizada no algoritmo permitia uma rápida ordenação por não-dominância, o que mantinha a diversidade entre as soluções e utilizava elitismo para preservar as melhores soluções ao longo das gerações. Para os experimentos, foi utilizada a topologia mesh para interligar switches e hosts. O algoritmo tinha a capacidade de explorar diversos caminhos, oferecendo respostas às demandas, e a topologia interligada proporciona um maior espaço de busca e mais recursos a serem explorados ([ASSIS; ERRICO, 2014](#)).

No artigo de [Silva \(2018\)](#), é apresentado um algoritmo que pode ser executado na mesma máquina física ou virtual do controlador, permitindo o compartilhamento de recursos computacionais. O algoritmo também pode ser executado em outro servidor, permitindo a separação lógica ou física do controlador e do algoritmo. A comunicação entre eles é realizada por meio de uma interface REST e de mensagens HTTP. O algoritmo monitora toda a rede com base nas informações fornecidas pelo controlador, identificando sobrecar-

gas e acionando um módulo de balanceamento de carga quando um equipamento atinge um percentual de utilização predefinido.

Em (SOUZA, 2014), foi realizada uma pesquisa com o objetivo de estudar os protocolos de roteamento da internet, como Routing Information Protocol - RIP, Open Shortest Path First - OSPF e BGP, bem como suas arquiteturas, a fim de implementar algoritmos de roteamento em SDN. A pesquisa fez uso de livros e artigos para analisar os algoritmos de roteamento usados na internet, incluindo o algoritmo de estado de enlace (LS), também conhecido como algoritmo de Dijkstra, e o algoritmo de vetor de distância.

Além disso, foram estudados os componentes de roteamento Inter-AS, suas características e funcionalidades, e como isso poderia ser aplicado em SDN. A pesquisa também abordou o protocolo BGP e a arquitetura RouterFlow, que utiliza um servidor externo para o roteamento IP, bem como o Inter-SDN, implementado como uma aplicação SDN. Foi destacado que a seleção de rotas do protocolo BGP se aplica aos princípios das SDN e que o processo de seleção de rotas segue os mesmos passos (SOUZA, 2014).

No trabalho de Pacífico *et al.* (2018), é apresentada uma proposta de um roteador SDN implementado em hardware. Essa implementação permite a introdução de novos campos e protocolos de forma dinâmica, eliminando a necessidade de recompilar ou reiniciar o roteador quando as configurações de processamento de fluxos são modificadas durante a execução. O roteador foi desenvolvido e testado em uma plataforma NetFPGA 1G em um ambiente real, demonstrando um alto poder de processamento de pacotes. Este roteador não possui um comportamento pré-definido, exceto por um processador de pacotes genérico. Em vez disso, seu comportamento é determinado por instruções que acessam os cabeçalhos dos pacotes e combinam ações dinamicamente durante a execução. Esse sistema faz uso do processador de pacotes eBPF¹.

No artigo de (TSAI; RISDIANTO; LING, 2022) foi verificado que O rápido crescimento da Internet tem dado origem ao conceito de "redes sob demanda", que ganha cada vez mais popularidade na gestão e operação de redes. No entanto, nas escolas de países em desenvolvimento, os recursos educacionais relacionados à SDN e Border Gateway Protocol (BGP) são limitados, e propôs a criação de um ambiente para a experimentação de tutoriais SDN e BGP. Nelse foi apresentado um projeto de desenvolvimento de laboratório, financiado pela ISIF Asia/APNIC Foundation, com o foco principal em estabelecer um ambiente de intercâmbio de Internet baseado em SDN.

Já no trabalho apresentado por Rezende *et al.* (2016), a proposta envolveu o desenvolvimento de um módulo multicaminho independente de protocolo, capaz de proporcionar

¹ O eBPF é uma tecnologia com originada no kernel do Linux, que possibilita a execução de programas em um ambiente isolado e privilegiado, como o próprio núcleo do sistema operacional. Essa tecnologia é empregada para estender com segurança e eficiência as funcionalidades do kernel, sem a necessidade de modificar o código-fonte do kernel ou carregar módulos do kernel.

suporte a multicaminho para todos os fluxos na rede Openflow, sem a necessidade de alterar os protocolos dos *hosts*.

Com base nos dados obtidos do plano de informações, fornecidos pelo módulo de monitoramento, o módulo multicaminho é capaz de efetuar automaticamente a determinação dos caminhos a serem seguidos. Ele realiza a programação do encaminhamento de pacotes em dispositivos do plano de dados, como *switches Openflow*, que já foram configurados de acordo com as decisões tomadas pelo próprio módulo multicaminho. O módulo de monitoramento, integrado ao plano de controle, assume a responsabilidade pelo processamento e armazenamento das estatísticas dos *switches*, em colaboração com o módulo multicaminho (REZENDE *et al.*, 2016).

O plano de aplicação também inclui duas aplicações, o Monitoramento e o Multicaminhos. Através da aplicação de Monitoramento, é possível observar todo o estado do plano de dados e como ele está em determinado momento, bem como ter acesso a informações sobre o consumo de recursos por meio de uma interface oferecida pela aplicação. Com a aplicação de Multicaminhos, os usuários podem contratar recursos de rede indicando a largura de banda para um determinado fluxo. O módulo de monitoramento processa e armazena estatísticas dos switches, e a recuperação da topologia de redes é feita a partir do Módulo de Gerenciamento de Topologia (REZENDE *et al.*, 2016).

O módulo de multicaminhos define os caminhos a serem usados para enviar os pacotes de dados. Os dados coletados pelo módulo de monitoramento são armazenados no Módulo de Armazenamento, onde várias informações e dados sobre a rede são mantidos (REZENDE *et al.*, 2016).

3.4 A SDN em soluções proprietárias

3.4.1 Redes Baseadas na Intenção

A Redes Baseada na Intenção (IBN, do inglês *Intention Based Network*) da Cisco se baseia em um modelo de rede SDN. Com foco na intenção da empresa em como conecta e protege seus usuários, dispositivos, aplicativos e cargas de trabalho em qualquer lugar. Com as informações e a automação habilitadas com o uso de Inteligência Artificial da Cisco, a Rede IBN aprende e se adapta continuamente às necessidades da empresa que a utiliza (CISCO, 2023).

As IBN possuem amplitude e profundidade para atender de forma completa as necessidades de uma rede. Ela abrange ambientes de acesso, WAN, data center e nuvem para organizações e indústrias de todos os tamanhos. (CISCO, 2023).

Ela fornece à equipe da empresa opções flexíveis de assinatura que permitem que um acompanhamento das crescentes demandas da sua rede à medida que as tecnologias e

requisitos mudam seja feita de maneira mais simples. Segundo (CISCO, 2023), os principais benefícios das Redes IBN são:

- Otimiza as experiências de aplicativos, garantindo o desempenho em redes privadas e públicas.
- Amplia para multi-cloud com conectividade segura para e entre cargas de trabalho distribuídas.
- Melhora a performance das operações de rede com plataformas baseadas na intenção que fornecem automação e informações centralizadas.
- Aprimora a segurança dos dados nas perspectivas dos pilares integridade, autenticidade, disponibilidade, confidencialidade. Protegendo usuários e aplicativos com acesso baseado em confiança zero.

A extensibilidade da plataforma aberta aumenta significativamente o valor da rede, permitindo a implantação de novas e inovadoras aplicações, endpoints, processos e soluções que são necessários no presente e que serão necessários no futuro (CISCO, 2023). Outras empresas desenvolvem suas soluções na área da SDN, um outro exemplo é a Ubiquiti.

3.4.2 A solução Ubiquiti Unifi

A integração da arquitetura SDN com a solução enterprise UniFi da Ubiquiti tem sido um tópico relevante na área de computação, apresentando um crescimento significativo nos últimos anos. Isso inclui a adoção de soluções comerciais por parte dos principais fabricantes de dispositivos de rede. É importante ressaltar que a solução UniFi® da Ubiquiti® é intrinsecamente baseada na arquitetura SDN desde seu lançamento comercial em 2010 (UBIQUITI, 2018).

A abordagem subjacente busca uma maior separação entre os planos de encaminhamento e de controle em dispositivos de rede, como roteadores, switches e pontos de acesso (APs). No modelo convencional, esses dispositivos operam com dois planos lógicos distintos (UBIQUITI, 2018):

- Plano de Encaminhamento: Responsável pela função primordial de direcionar pacotes entre interfaces.
- Plano de Controle e Gerenciamento: Encarregado de gerir configurações, protocolos e a tomada de decisões inteligentes nos dispositivos.

Essa abordagem visa aprofundar a separação entre esses dois planos, aumentando a flexibilidade e a eficiência na operação de redes de comunicação.

O SDN introduz uma ruptura nesse modelo tradicional, com o desacoplamento ainda maior entre os planos de controle e encaminhamento. Controladores programáveis por software são adicionados à rede, assumindo a responsabilidade pelo controle e inteligência. Esses controladores podem ser programados de forma flexível, permitindo o gerenciamento e até mesmo a reconfiguração da rede de acordo com as necessidades reais. A solução UniFi da Ubiquiti adota essa abordagem, e a figura abaixo ilustra essa topologia de rede baseada nos produtos UniFi, com destaque para a controladora (UBIQUITI, 2018).

Essa abordagem simplifica os dispositivos de infraestrutura, que agora focam exclusivamente no encaminhamento de pacotes, desde que possuam interfaces programáveis por software (APIs) para se comunicar com os controladores externos. A Ubiquiti oferece a solução SDN por meio do software UniFi Controller, que é gratuito e simplifica a configuração e monitoramento de todos os ativos UniFi na rede. A interface gráfica (GUI) substitui a complexidade da programação e da configuração via linha de comando (CLI), mantendo a abordagem de simplificação da solução UniFi (UBIQUITI, 2018).

O UniFi Controller permite a configuração centralizada de todos os equipamentos na infraestrutura da rede, que são posteriormente provisionados pelos dispositivos. Uma característica notável é que a operação dos dispositivos não depende estritamente da controladora, embora seja recomendado para obter maior visibilidade da rede. Diferentemente de outros concorrentes, a Ubiquiti evita a necessidade de soluções complexas e caras com equipamentos redundantes para manter a operação da rede em caso de falha da controladora.

De acordo com UBIQUITI (2018), uma única controladora UniFi tem a capacidade de administrar múltiplos sites independentes, criando uma interface dedicada para cada um deles. Além disso, é possível fazer o upload de mapas para visualizar diferentes áreas e monitorar a intensidade do sinal dos pontos de acesso. O UniFi foi projetado com o propósito de possibilitar a gestão remota dos pontos de acesso, mesmo em redes geograficamente distantes, por meio de técnicas de adoção em Layer-3, contanto que haja conectividade entre essas redes.

O software da controladora UniFi é disponibilizado gratuitamente e pode ser instalado em diversas infraestruturas, incluindo servidores físicos, máquinas virtuais na nuvem e até mesmo em computadores pessoais. Além disso, de acordo com UBIQUITI (2018), o UniFi Cloud-Key representa uma alternativa econômica que também proporciona acesso remoto por meio da nuvem.

Resumidamente, os produtos pertencentes à família Ubiquiti UniFi oferecem uma solução abrangente para a administração de redes a um preço competitivo, sem a necessidade de aquisição de licenças de software.

3.4.3 Abordagem da VMware em SDN

Para [VMWare \(2023\)](#) a atual paisagem tecnológica, caracterizada pela proliferação de serviços e aplicativos, especialmente aqueles envolvendo computação em nuvem, é amplamente dependente da Rede Definida por Software (SDN).

A SDN desempenha um papel fundamental na capacidade de mover dados eficientemente entre locais distribuídos, tornando-se um elemento essencial para o funcionamento eficaz de aplicativos baseados em nuvem. Além disso, a SDN oferece suporte à rápida mobilidade das cargas de trabalho através da rede, possibilitando, por exemplo, a segmentação de redes virtuais por meio da virtualização das funções de rede (NFV), permitindo que os provedores de telecomunicações redimensionem os serviços de atendimento ao cliente de forma eficiente. Isso, por sua vez, capacita os provedores de serviços a migrar cargas de trabalho entre infraestruturas em nuvem privada e pública conforme necessário, disponibilizando novos serviços de atendimento ao cliente instantaneamente.

A VMware, conhecida por criar soluções de virtualização e computação em nuvem, oferece uma série de produtos e recursos relacionados à Rede Definida por Software (SDN), suas soluções vêm a seguir:

VMware NSX: Software de Virtualização de Redes e Segurança

O VMware NS X[®] Data Center representa uma solução de ponta em termos de rede virtualizada e segurança totalmente baseada em software. Este software inovador oferece a flexibilidade necessária para implementar redes virtualizadas de maneira eficiente e segura, tornando-se um componente crucial na infraestrutura moderna de TI ([VMWARE, 2023](#)).

VMware SD-WAN: Simplificando a Rede da Filial

A rede remota definida por software (SD-WAN) da VMware simplifica significativamente as operações de rede das filiais, otimizando o desempenho de aplicativos que dependem de conexões pela Internet e redes remotas híbridas. Esta solução permite a gestão eficaz da conectividade em ambientes distribuídos, resultando em melhorias substanciais na experiência do usuário e na eficiência operacional ([VMWARE, 2023](#)).

Infraestrutura Convergente e sua Importância

A infraestrutura convergente emerge como uma estratégia de TI poderosa, integrando servidores, armazenamento, rede e software de gerenciamento em uma solução predefinida. A VMware compreende a relevância dessa abordagem e oferece soluções de infraestrutura convergente para atender às crescentes demandas do mercado de TI ([VMWARE, 2023](#)).

Data Center Definido por Software: Uma Visão Detalhada

A virtualização de servidores x86, introduzida pela VMware há mais de uma década, tornou-se uma tecnologia amplamente adotada em data centers em todo o mundo. Essa tecnologia revolucionária possibilitou a criação de ambientes de data center altamente vir-

tualizados e eficientes, desempenhando um papel crucial na evolução da infraestrutura de TI (VMWARE, 2023).

VMware vRealize Network Insight: Otimizando Infraestruturas de Rede Multi-Cloud

O VMware vRealize Network Insight é uma solução que capacita os clientes a criar infraestruturas de rede altamente disponíveis, seguras e otimizadas em ambientes multi-cloud. Essa ferramenta fornece visibilidade e controle essenciais para gerenciar as complexidades das redes modernas, garantindo um desempenho confiável e seguro (VMWARE, 2023).

Nuvem Privada Definida por Software: Modelando o Futuro dos Negócios

A nuvem privada definida por software oferece uma base digital unificada para atender às demandas futuras dos negócios. Com sua capacidade de oferecer modelos operacionais consistentes e flexíveis, a nuvem privada definida por software permite que as organizações enfrentem os desafios em constante evolução do mercado de maneira eficaz e eficiente (VMWARE, 2023).

Virtual Cloud Network: A Rede do Futuro

O Virtual Cloud Network, baseado na tecnologia VMware NSX, representa a visão da VMware para a rede do futuro. Essa camada de software onipresente abrange desde o data center até a nuvem e a infraestrutura de borda, fornecendo uma plataforma de rede altamente adaptável e segura (VMWARE, 2023).

A análise das soluções de SDN da VMware revela uma ampla gama de recursos e produtos que capacitam as organizações a enfrentar os desafios da moderna infraestrutura de TI. A capacidade de virtualizar redes, otimizar o desempenho, simplificar operações e promover a flexibilidade é fundamental para atender às crescentes demandas da era digital. A VMware demonstra liderança contínua no campo da SDN, contribuindo significativamente para a evolução das redes de computadores e da infraestrutura de TI como um todo. Esta análise serve como um panorama abrangente das soluções de SDN da VMware e sua importância no cenário tecnológico atual.

Os trabalhos relacionados indicam que a área de Redes SDN não apenas se estabeleceu no mercado, mas também continua a ser um campo de pesquisa acadêmica ativo e em constante evolução. Enquanto as soluções empresariais têm seu foco nas demandas corporativas, os estudos acadêmicos sobre SDN são prolíficos e abrangem uma variedade de tópicos. Para obter informações mais detalhadas sobre essas pesquisas, é recomendável consultar diretamente os trabalhos acadêmicos citados nos artigos mencionados. Isso oferecerá uma compreensão mais aprofundada das inovações e avanços em curso na área de Redes Definidas por Software.

3.5 Como Implementar Lógica Fuzzy utilizando Python e a Mininet

Neste estudo de caso, foi selecionado como problema de pesquisa a variação de latência de rede, comumente referida como "*jitter*". A análise de métricas de desempenho, como latência, perda de pacotes e jitter², desempenha um papel crucial no ambiente de redes corporativas por várias razões significativas. Primeiramente, é essencial garantir a qualidade de serviço (QoS) dos serviços oferecidos pela rede. Em redes corporativas, que frequentemente suportam serviços críticos, como voz sobre IP (VoIP), videoconferência e transferência de arquivos, métricas precisas de desempenho são fundamentais para fornecer uma experiência de alta qualidade aos usuários. A latência, a perda de pacotes e o jitter podem afetar diretamente a qualidade e a confiabilidade desses serviços (ITU-T, 2008).

Além disso, a análise das métricas de desempenho é uma prática indispensável para identificar gargalos e problemas na rede corporativa. Taxas elevadas de perda de pacotes podem indicar congestionamento ou falhas em dispositivos de rede, enquanto a latência excessiva pode ser um sinal de problemas de roteamento ou capacidade inadequada nos links. O jitter pode indicar instabilidade na rede, prejudicando a entrega consistente dos pacotes. Ao monitorar e analisar essas métricas, os administradores de rede podem diagnosticar e solucionar problemas, garantindo um funcionamento eficiente da rede (TANENBAUM; WETHERALL, 2021).

Adicionalmente, a análise de métricas de desempenho auxilia no planejamento de capacidade e dimensionamento da rede. Ao obter *insights* sobre a utilização atual da rede e identificar pontos de gargalo (estreitamento da largura de banda), as empresas podem planejar a capacidade e dimensionar adequadamente os recursos de rede. Isso garante que a rede esteja preparada para lidar com o crescimento futuro e as demandas dos serviços e aplicativos corporativos.

Outro aspecto relevante é a melhoria da experiência do usuário. Métricas de desempenho otimizadas, como baixa latência, perda de pacotes mínima e jitter controlado, resultam em uma experiência de rede mais rápida e estável. Isso é fundamental para a satisfação dos usuários e para o desempenho eficiente das atividades corporativas.

Ademais, a análise de métricas de desempenho também está intimamente relacionada ao monitoramento e à conformidade com acordos de nível de serviço (SLAs)³. Muitas redes corporativas possuem SLAs estabelecidos com clientes internos ou externos, que especificam requisitos de desempenho. Ao monitorar e analisar essas métricas, as empresas

² Jitter é uma métrica utilizada para descrever a variação no atraso de pacotes em uma rede de comunicação. Em outras palavras, é a diferença de tempo entre a chegada de pacotes em um destino, considerando a sua taxa média de chegada.

³ Documento formal em que duas empresas definem o que uma deve fornecer e o que a outra pode exigir da primeira, em uma relação profissional

podem garantir a conformidade com os SLAs, assegurando que os serviços de rede atendam aos requisitos acordados.

3.5.1 Implementação

Para implementar um sistema com lógica Fuzzy em Python para analisar métricas de desempenho, latência, perda de pacotes e jitter, pode-se empregar a biblioteca de lógica Fuzzy ‘scikit-fuzzy’.

No trecho de código 3.1 é mostrado uma implementação do sistema.

Neste exemplo, definiu-se três variáveis de entrada: *latencia*, *perda_pacotes* e *jitter*, e uma variável de saída *desempenho*. Em seguida, definiu-se as funções de pertinência para cada uma dessas variáveis, utilizando a função *trimf* da biblioteca *scikit-fuzzy*.

Após definir as funções de pertinência, criamos regras Fuzzy para relacionar as variáveis de entrada com a variável.

Para integrar o código de lógica Fuzzy em um ambiente Mininet utiliza-se eventos ou informações de monitoramento de rede fornecidas pelo Mininet para alimentar o sistema de controle Fuzzy e usar a saída desse sistema para tomar decisões relacionadas à rede.

No Código 3.2, é apresentado um exemplo simples de como é possível incorporar o código de lógica Fuzzy em um ambiente Mininet.

Após se define a função de evento *monitor_network* que é acionada pelo Mininet para monitorar as métricas de desempenho, latência, perda de pacotes e jitter. Dentro dessa função de evento, se atualizam as entradas do sistema de controle Fuzzy com as métricas obtidas, computa-se o resultado e toma-se decisões com base no valor de saída.

O evento de monitoramento de rede é adicionado ao switch (ou host relevante no Mininet) para que o evento seja acionado durante a simulação. Neste exemplo, o evento foi adicionado ao switch1 usando o método *monitor*. Por fim, inicia-se a rede Mininet, executa-se a simulação ou experimento e, em seguida, para e limpa a topologia do Mininet.

3.5.2 Projetos públicos sob SDN e QoS

As pesquisas acadêmicas e as aplicações desenvolvidas por empresas no campo de SDN (Redes Definidas por Software) e QoS (Qualidade de Serviço) desempenham um papel crucial na evolução e na melhoria das redes de computadores.

Tanto as pesquisas acadêmicas quanto as aplicações empresariais em SDN e QoS têm como objetivo melhorar a eficiência, a flexibilidade e a qualidade das redes de computadores. Essa convergência de pesquisa e implementação prática está moldando o futuro das comunicações e da infraestrutura de rede em todo o mundo.

Código 3.1 – Código Python análise métrica

```
1 import numpy as np
2 import skfuzzy as fuzz
3 from skfuzzy import control as ctrl
4
5 # Definindo as variaveis de entrada
6 latencia = ctrl.Antecedent(np.arange(0, 101, 1), 'latencia')
7 perda_pacotes = ctrl.Antecedent(np.arange(0, 101, 1), 'perda_pacotes')
8 jitter = ctrl.Antecedent(np.arange(0, 101, 1), 'jitter')
9
10 # Definindo as variaveis de saida
11 desempenho = ctrl.Consequent(np.arange(0, 101, 1), 'desempenho')
12
13 # Definindo as funcoes de pertinencia para cada variavel
14 latencia['baixa'] = fuzz.trimf(latencia.universe, [0, 0, 50])
15 latencia['alta'] = fuzz.trimf(latencia.universe, [0, 50, 100])
16
17 perda_pacotes['baixa'] = fuzz.trimf(perda_pacotes.universe, [0, 0, 50])
18 perda_pacotes['alta'] = fuzz.trimf(perda_pacotes.universe, [0, 50, 100])
19
20 jitter['baixo'] = fuzz.trimf(jitter.universe, [0, 0, 50])
21 jitter['alto'] = fuzz.trimf(jitter.universe, [0, 50, 100])
22
23 desempenho['bom'] = fuzz.trimf(desempenho.universe, [0, 0, 50])
24 desempenho['ruim'] = fuzz.trimf(desempenho.universe, [0, 50, 100])
25
26 # Regras Fuzzy
27 rule1 = ctrl.Rule(latencia['baixa'] & perda_pacotes['baixa'] & \
28 jitter['baixo'], desempenho['bom'])
29 rule2 = ctrl.Rule(latencia['alta'] | perda_pacotes['alta'] \
30 jitter['alto'], desempenho['ruim'])
31
32 # Sistema de controle Fuzzy
33 desempenho_ctrl = ctrl.ControlSystem([rule1, rule2])
34 desempenho_simulador = ctrl.ControlSystemSimulation(desempenho_ctrl)
35
36 # Definindo os valores de entrada
37 desempenho_simulador.input['latencia'] = 20
38 desempenho_simulador.input['perda_pacotes'] = 30
39 desempenho_simulador.input['jitter'] = 10
40
41 # Computando o resultado
42 desempenho_simulador.compute()
43
44 # Obtendo o valor de saida
45 print(desempenho_simulador.output['desempenho'])
```

Código 3.2 – Lógica Fuzzy e Mininet

```

1 from mininet.net import Mininet
2 from mininet.node import Controller, OVSKernelSwitch
3 from mininet.link import TCLink
4 import numpy as np
5 import skfuzzy as fuzz
6 from skfuzzy import control as ctrl
7
8 # Codigo de definicao do sistema de controle Fuzzy
9
10 # Criacao da topologia de rede no Mininet
11 net = Mininet(controller=Controller, switch=OVSKernelSwitch, link=TCLink
12 )
13
14 # Adicione hosts, switches e links aqui
15 # Defina uma funcao de evento para monitorar metricas\
16 # de desempenho, latencia, perda de pacotes e jitter
17
18 def monitor_network(event):
19     # Obtenha as metricas de desempenho, latencia,\
20     # perda de pacotes e jitter
21     # Atualize as entradas do sistema de controle\
22     # Fuzzy com as metricas obtidas
23
24     desempenho_simulador.input['latencia'] = latencia_value
25     desempenho_simulador.input['perda_pacotes'] = perda_pacotes_value
26     desempenho_simulador.input['jitter'] = jitter_value
27
28     # Compute o resultado do sistema de controle Fuzzy
29     desempenho_simulador.compute()
30     # Obtenha o valor de saida
31     desempenho = desempenho_simulador.output['desempenho']
32
33 # Tome decisoes com base no valor de saida, como
34 # ajustar politicas de roteamento, priorizacao de trafego, etc.
35 # Adicione o evento de monitoramento de rede ao Mininet
36 net.get('switch1').monitor(monitor_network)
37
38 # Inicialize a rede Mininet
39 net.start()
40 # Execute sua simulacao ou experimento no Mininet
41 # Pare a simulacao e limpe a topologia do Mininet
42 net.stop()

```

Há uma gama de projetos disponíveis nos repositórios do GitHub que apresentam abordagens e implementações relacionadas à Qualidade de Serviço (QoS) em Redes Definidas por Software (SDN), cabendo destacar os seguintes.

No repositório *amirashoori7/sdn_qos*, é possível encontrar um conjunto de recursos e códigos que visam aprimorar o gerenciamento de QoS em ambientes SDN (ASHOORI, 2020).

No repositório *joagonzalez/sdn_qos*, pode-se explorar soluções relacionadas à QoS em SDN, com foco na otimização do desempenho da rede e na garantia de uma experiência de usuário de alta qualidade (GONZALEZ, 2020).

Por fim, o repositório *Gradiant/SDN-QoS-PoC* se concentra em apresentar um conceito de prova de conceito (PoC) relacionado à QoS em SDN, demonstrando a viabilidade de suas soluções em cenários práticos ([GALICIA, 2020](#)).

Esses projetos oferecem recursos e implementações no campo da QoS em SDN, contribuindo para o desenvolvimento e aprimoramento de soluções nesse domínio.

Capítulo 4

Determinação de Largura de Banda com Lógica Fuzzy

A pesquisa prática realizada nesta dissertação foi publicada. Aqui foi criado um protótipo em Python que utiliza lógica Fuzzy para calcular a alocação proporcional de largura de banda para cada usuário, com o objetivo de preservar a Qualidade de Serviço (QoS) na distribuição da velocidade máxima entre os usuários. (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

A lógica Fuzzy (lógica nebulosa) é baseada na lógica multi-valorada e trabalha com a inserção de variáveis linguísticas e suas incertezas formando os conjuntos difusos (nebulosos). Sistemas computacionais que precisam lidar com incertezas podem utilizar a lógica fuzzy para resolver esse tipo de problema (SANDRI; CORREA, 1999).

Devido ao grande volume de trabalho na administração de Redes de Computadores, um software que auxilie em alguma das tarefas é desejável. Como o auxílio no controle da velocidade máxima que a conexão de um usuário pode chegar em uma rede de computadores. Essa tarefa possui uma série de variáveis que só podem ser corretamente inseridas no sistema se modeladas com suas incertezas.

Nesse contexto, a lógica Fuzzy é uma ferramenta que é capaz de dar provimento à modelagem de tal sistema. Aqui serão exibidos os detalhes da criação de um protótipo de aplicação feito em Python para a resolução do problema da definição da largura de banda máxima para um usuário de redes de computadores e em seguida os resultados obtidos.

A lógica Fuzzy consegue, a partir de entradas subjetivas, produzir saídas determinísticas que podem ser usadas em decisões computacionais. Em um sistema gerenciador de usuários, por exemplo, precisa-se minerar a informação dos *logs*¹ de serviços utilizados.

O controle de banda baseado no comportamento dos usuários da rede é interessante

¹ Arquivos de texto gerados automaticamente por serviços, como serviços de Redes de Computadores, com informações sobre seu funcionamento e utilização.

do ponto de vista do administrador da rede, uma vez que regras de firewall e proxy não fazem a contenção de uso para toda situação que possa ocorrer. Um controle de banda automatizado e inteligente pode ajudar a manter a rede disponível da melhor maneira possível para seus usuários. Pensando nisso, este estudo de caso mostra um protótipo simplificado de controle da banda máxima disponibilizada a um usuário da rede utilizando lógica Fuzzy.

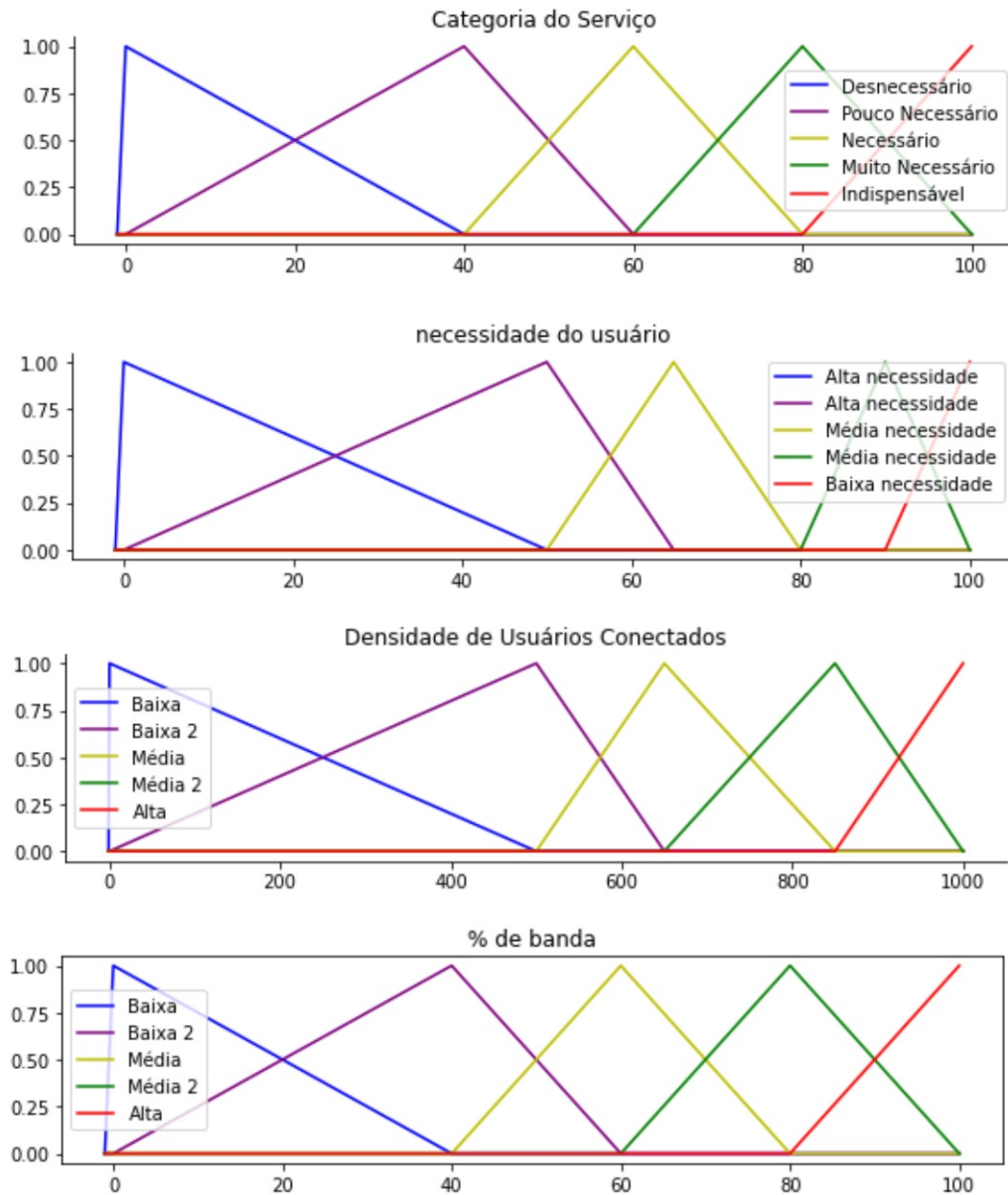
Foram designadas 3 variáveis de entrada, com 5 estados possíveis, pensando nos requisitos de QoS, como o mostrado por (TANENBAUM; WETHERALL, 2021) e (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022), sendo conjuntos nebulosos baseados em:

- **Categoria de Serviço:** Liberação parcial da velocidade máxima baseada na classificação da categoria de serviço de rede ou website utilizado. Quanto maior o valor da prioridade maior deve ser o impacto positivo na largura de banda disponibilizada. Este possui uma função de pertinência distribuída de forma mais uniforme. Sendo 0 a liberação de nenhuma banda nesta categoria de entrada e 100 a liberação total da banda.
- **Necessidade do usuário:** baseadas no tipo de tarefa e intensidade de uso: Baseado na utilização necessidade de banda, o usuário A que precisa de mais conexão é o que trabalha com um maior volume de dados, ele precisa de mais velocidade para que seu trabalho demore o menos possível. Todavia um usuário B que trabalha com um menor volume de dados deve ter seu trabalho concluído mais rápido que o usuário A, para, por exemplo, poder iniciar um novo trabalho. Este fator visa um equilíbrio de liberação de banda máxima de acordo com a necessidade medida pelo volume de trabalho e para isso a função de pertinência vai ficando com intervalos menores a media que a classificação da necessidade do usuário vai aumentando, tendo um usuário com menos carga de trabalho um acesso a uma boa parte de velocidade máxima, mas não tão grande quanto o usuário com um volume de trabalho maior. Sendo 0 uma prioridade de necessidade mais baixa e 100 a mais alta. Neste caso a prioridade baixa tem seu efeito até a 50% dos valores possíveis, enquanto a prioridade mais alta tem seu efeito iniciando a 90%, assim mantendo o equilíbrio citado a cima.
- **Densidade de Usuários Conectados:** Consiste no controle da velocidade máxima(% de banda) para se adaptar à sobrecarga. Percebendo o volume de densidade de usuários na rede como um fator de sobrecarga da mesma, a banda liberada neste fator depende da quantidade de usuários. Neste estudo de caso, o número de usuários pode variar de 0 a 1000, sendo que o fator de sobrecarga é acrescido de forma mais significativa a partir de 500 (50%) usuários simultâneos conectados. Nesta variável, um maior valor significa menos banda disponível e seu efeito é inverso aos das outras duas variáveis.

A variável de saída é um conjunto nebuloso composto por 5 estados distintos, categorizados desde velocidade baixa até velocidade alta. Essa variável define a velocidade máxima

de banda que um usuário pode utilizar na rede. As funções de pertinência definidas para este estudo de caso são do tipo triangular e podem ser visualizadas na 4.1.

Figura 4.1 – Funções de Pertinências do Protótipo do Sistema Fuzzy



Fonte: O autor, em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

Para este estudo de caso, os dados crisp de entrada serão inseridos manualmente para verificar as saídas produzidas pela aplicação, todavia são informações que podem ser obtidas através de relatórios de serviços de rede. A aplicação foi desenvolvida em Python e utiliza as bibliotecas presentes nos pacotes Numpy, Skfuzzy e Matplotlib. A Numpy possui funções matemáticas avançadas e será utilizada para as operações nos conjuntos nebulosos e na definição dos valores de seus intervalos. A Skfuzzy vai ser usada para definir os elementos da lógica Fuzzy como as funções de pertinência, a fuzzificação e a defuzzificação. A Matplotlib

é utilizada para a visualização dos valores nebulosos em formato gráfico.([RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022](#))

O processo de criação do código inicia com a instalação das bibliotecas. Assim que prontas para uso elas devem ser importadas, em seguida se define os conjuntos nebulosos das variáveis de entrada e as respectivas funções de pertinência. Na sequência, os dados de entrada nítidos são submetidos ao processo de fuzzificação. Em um terceiro estágio, as regras de inferência são formuladas e aplicadas a cada variável de entrada, com a utilização da operação de "OU MÁXIMO" na definição das regras de cada variável.

Neste ponto do processo, estão sendo estabelecidas as associações entre as variáveis de entrada, sendo a Densidade de Usuários Conectados inversamente associada à Categoria de Serviços e à Necessidade do Usuário. Isso define o conjunto de regras para a variável linguística de saída.

No quarto estágio, a inferência é realizada, combinando todas as regras definidas. Em seguida, ocorre a etapa de defuzzificação, na qual o ponto central da variável de saída é utilizado. Isso resulta em um valor em ponto flutuante² que representa o percentual de banda recomendado para o usuário.

As regras da base de conhecimento, conforme em ([RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022](#)) podem ser vistas a seguir:

- Categoria do Serviço
 - Se Categoria do serviço for muito fraca, então, a velocidade máxima de conexão será baixa.
 - Se Categoria do serviço for fraca, então, a velocidade máxima de conexão será média baixa.
 - Se Categoria do serviço for mediana, então, a velocidade máxima de conexão será mediana.
 - Se Categoria do serviço for média alta, então, a velocidade máxima de conexão será média alta.
 - Se Categoria do serviço for alta, então, a velocidade máxima de conexão será alta.
- Necessidade do Usuário
 - Se Necessidade do Usuário for muito fraca, então, a velocidade máxima de conexão será média baixa.

² Números com casas decimais.

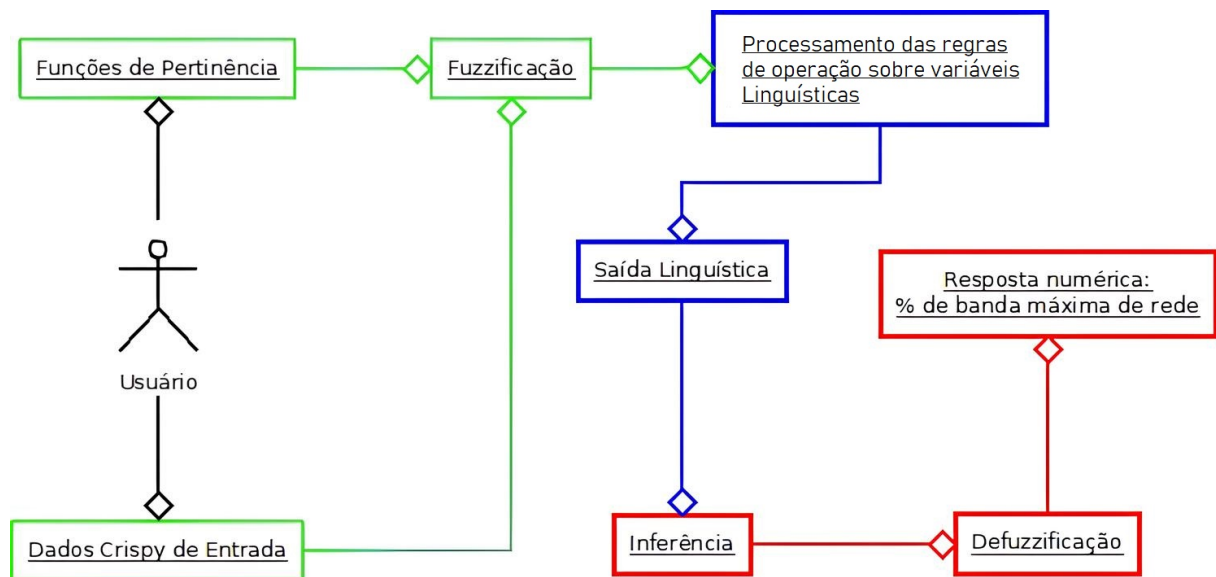
- Se Necessidade do Usuário for fraca, então, a velocidade máxima de conexão será mediana.
 - Necessidade do Usuário do serviço for mediana, então, a velocidade máxima de conexão será média alta.
 - Se Necessidade do Usuário for média alta, então, a velocidade máxima de conexão será média alta.
 - Se Necessidade do Usuário for alta, então, a velocidade máxima de conexão será muito alta.
- Densidade de Usuários Conectados
 - Se Densidade de Usuários Conectados for muito baixa, então, a velocidade máxima de conexão será muito alta.
 - Se Densidade de Usuários Conectados for baixa, então, a velocidade máxima de conexão será alta.
 - Se Densidade de Usuários Conectados for mediana, então, a velocidade máxima de conexão mediana.
 - Se Densidade de Usuários Conectados for média alta, então, a velocidade máxima de conexão mediana baixa.
 - Se Densidade de Usuário Conectados for alta, então, a velocidade máxima de conexão baixa.

Do ponto de vista do usuário, o processamento de dados tem início na fase de fuzzificação, considerando que as funções de pertinência já estão previamente definidas. No entanto, é importante destacar que essas funções também podem ser ajustadas pelo próprio usuário, proporcionando flexibilidade no sistema.

Para ilustrar a dinâmica da execução desse protótipo, foi desenvolvido um diagrama de caso de uso, conforme representado na Figura 4.2. Esse diagrama esquematiza as interações e as possíveis ações que um usuário pode realizar ao utilizar o sistema, fornecendo uma visão clara do funcionamento do protótipo.

O código implementado está disponível no Apêndice A(organizado para o trabalho escrito) e no Google Colab(original) no link disponível em: <https://colab.research.google.com/drive/13mbAXe3SIsoWqLhMr_oXXg6YMtgd83Km>

Figura 4.2 – Diagrama de Caso de Uso do Sistema Protótipo



Fonte: O autor, em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

4.1 Saídas e Avaliação dos Resultados

Foram realizados diferentes experimentos com entradas diversas. A seguir serão mostrados 4 destes experimentos:

Experimento A: serviços com prioridade baixa de 10%, Usuário de prioridade de uso média com 50% e rede sobrecarregada com 980 usuários ativos. As saídas podem ser vista na figura 4.3.

No experimento A, o usuário terá um total de 26.60% da banda de download e upload disponível como velocidade máxima de conexão de rede. Percebe-se que em uma situação em que o usuário está com pouco privilégio e com a rede sobrecarregada a velocidade máxima que ele pode atingir é baixa.

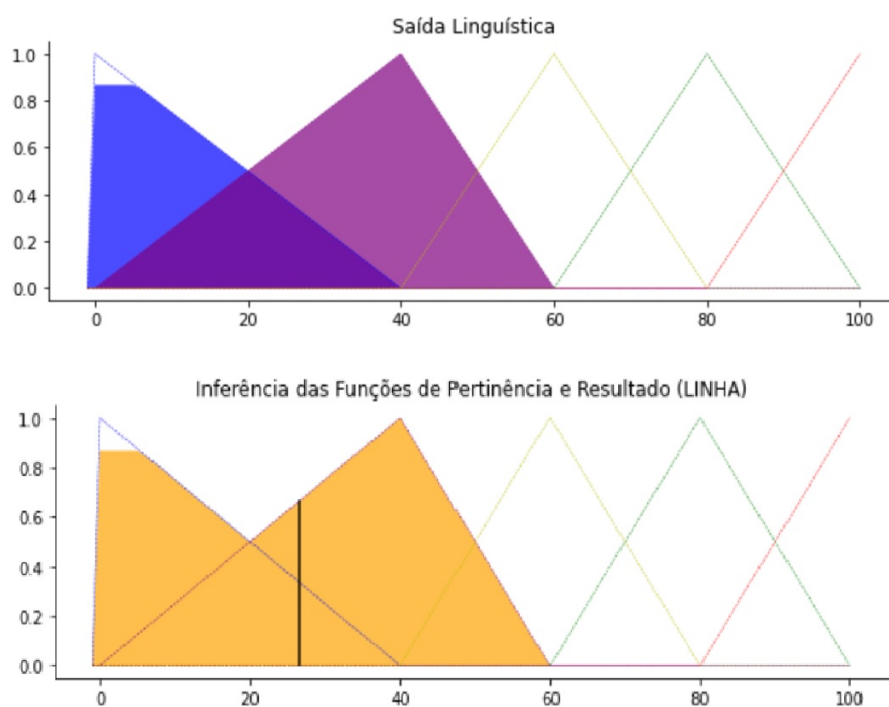
Experimento B: Tem como base o experimento A, mas sem uma sobrecarga significativa no número de usuários que possui 80 usuários ativos. As saídas podem ser vistas na figura 4.4.

No experimento B, o usuário terá um total de 42.24% da banda de download e upload disponível como velocidade máxima de conexão de rede. Percebe-se que em uma situação em que o usuário está com pouco privilégio e com a rede livre a velocidade máxima que ele pode atingir é mediana.

No experimento C, serviços com prioridade alta de 100%, Usuário de prioridade de uso alta com 100% e rede sobrecarregada com 1000 usuários ativos. As saídas podem ser vista na figura 4.5.

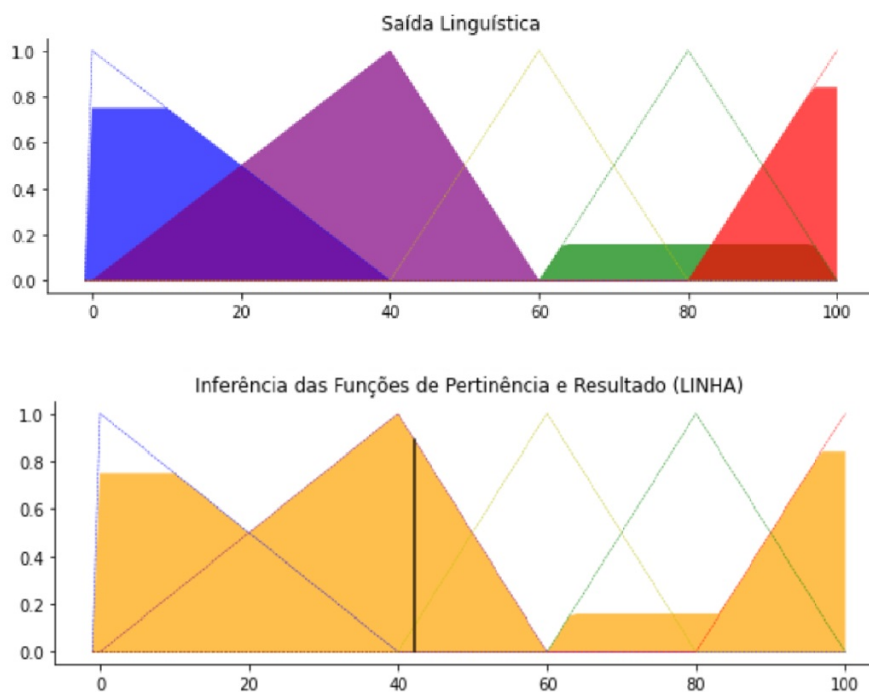
No experimento C, o usuário terá um total de 39.34% da banda de download e upload

Figura 4.3 – Saídas do Experimento A



Fonte: O autor, em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

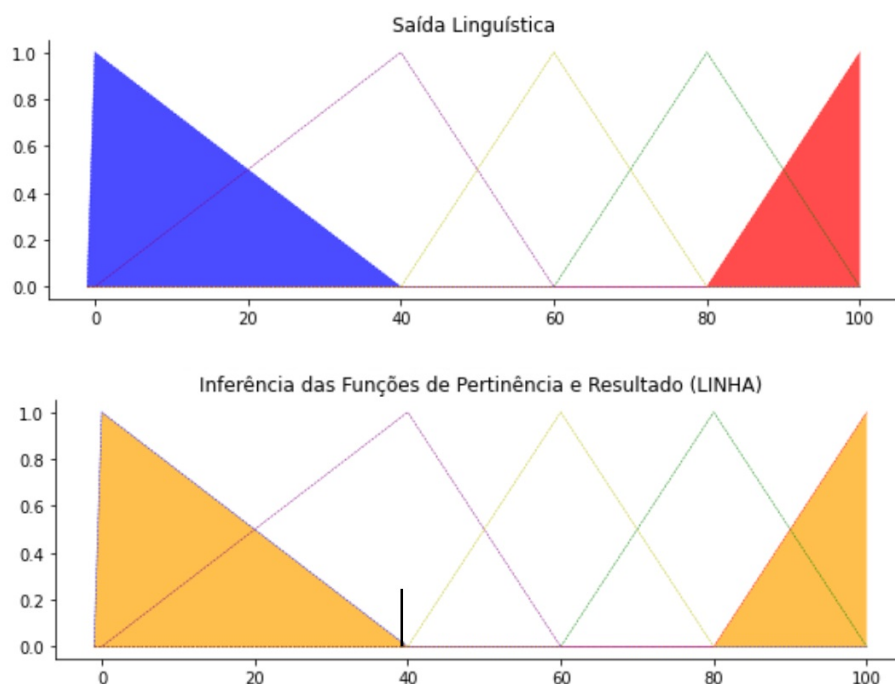
Figura 4.4 – Saídas do Experimento B



Fonte: O autor, em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

disponível como velocidade máxima de conexão de rede. Percebe-se que em uma situação em que o usuário está com privilégio muito alto e com a rede extremamente sobrecarregada

Figura 4.5 – Saídas do Experimento C



Fonte: O autor, em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

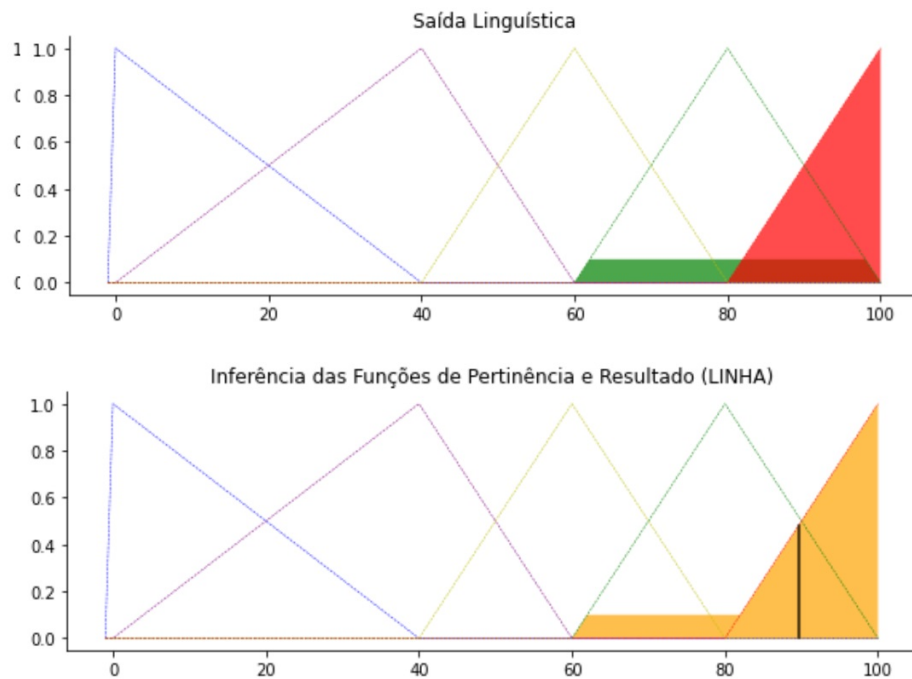
a velocidade máxima que ele pode atingir é mediana baixa. O que mostra o equilíbrio alcançado pela saída do protótipo que mesmo sendo uma velocidade máxima percentual baixa, mediante o grande número de usuário é um percentual bom e na prática nem sempre será atingido devido a alta demanda.

Experimento D: Tem como base o experimento C, mas sem uma sobrecarga significativa no número de usuários que possui 50 usuários ativos. As saídas podem ser vistas na figura 4.6.

No experimento D, o usuário terá um total de 89.61% da banda de download e upload disponível como velocidade máxima de conexão de rede. Percebe-se que em uma situação em que o usuário está com privilégio muito alto e com a rede com baixa sobrecarregada a velocidade máxima que ele pode atingir é alta. Considerando que existem outros 49 usuários, dificilmente ele vai atingir esse percentual de velocidade, mas o limite definido é interessante e expressa o equilíbrio alcançado pela saída do sistema ajudando no controle de banda para uma distribuição mais justa de velocidade para todos os usuários.

Os experimentos realizados mostram como as variáveis de entrada influenciam diretamente na saída do sistema. Ficou claro que, mesmo com um número relativamente pequeno de variáveis de entrada, o sistema é capaz de realizar uma distribuição eficiente da velocidade máxima. É fundamental notar que essa velocidade máxima representa um limite superior, uma referência, e não necessariamente a velocidade exata que um usuário atin-

Figura 4.6 – Saídas do Experimento D



Fonte: O autor, em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022)

girá em todos os momentos. Este aspecto é especialmente relevante em ambientes de rede dinâmicos, onde a demanda pode variar consideravelmente.

O controle de banda máximo desempenha um papel crucial nesse contexto. Ele proporciona uma maneira automatizada e justa de gerenciar o acesso à rede. Esse controle, embora não garanta que todos os usuários alcancem a velocidade máxima o tempo todo, assegura que nenhum usuário monopolize indevidamente os recursos da rede, garantindo assim uma experiência mais equitativa para todos.

Ao estabelecer esse limite superior, os administradores de rede têm a capacidade de manter o desempenho da rede dentro de limites aceitáveis, mesmo em períodos de alta demanda. Isso é fundamental para garantir que todos os usuários tenham acesso aos recursos da rede, evitando gargalos e congestionamentos.

O controle de banda máximo emerge como uma ferramenta para os gerentes de rede, permitindo-lhes manter um equilíbrio entre a otimização de recursos e a garantia de uma experiência justa para todos os usuários. Os experimentos realizados destacam como essa abordagem pode ser eficaz e fornecer um controle automatizado que beneficia tanto os administradores quanto os usuários da rede.

Capítulo 5

CONCLUSÃO E TRABALHOS FUTUROS

A administração de redes de computadores desempenha um papel crítico e complexo nos ambientes tecnológicos contemporâneos. O aumento constante na geração de dados em tempo real, incluindo relatórios de sistemas de gerenciamento de rede, sistemas e serviços, intensifica ainda mais a complexidade dessa tarefa. Nesse contexto, os softwares projetados para auxiliar os administradores de redes desempenham um papel importante, proporcionando ferramentas e recursos que simplificam a gestão e a otimização desses ambientes complexos.

A análise das saídas do protótipo, conforme documentado em (RIBEIRO JÚNIOR; GOMES JÚNIOR; MOURA JÚNIOR, 2022), revelou uma coerência entre as entradas e saídas do sistema. Essa congruência entre os dados de entrada e os resultados obtidos representa um indicativo do cumprimento do protótipo em seu objetivo de gerenciar o acesso à rede com base na prioridade do usuário.

O estudo realizado proporcionou uma observação da capacidade do sistema em adaptar-se às condições de congestão da rede ou à disponibilidade de recursos, permitindo tanto aos usuários com baixa prioridade quanto aos de alta prioridade acessar a rede com velocidades diferenciadas. Esse nível de flexibilidade é importante em ambientes nos quais a Qualidade de Serviço (QoS) desempenha um papel crítico. Ele permite que os usuários com maior prioridade possam receber tratamento preferencial, enquanto, ao mesmo tempo, os demais usuários ainda desfrutem de serviços aceitáveis, mesmo durante períodos de alta demanda.

A constatação positiva da coerência entre as entradas e saídas do protótipo sublinha a relevância fundamental do desenvolvimento de sistemas de gerenciamento de rede orientados por prioridades. Essa abordagem se estabelece como uma resposta eficaz aos desafios de assegurar uma experiência de usuário otimizada em ambientes de rede dinâmicos e frequentemente congestionados. Além disso, os resultados desta pesquisa estabelecem uma

base para investigações e implementações futuras no domínio do gerenciamento de redes e Qualidade de Serviço (QoS).

Um dos aspectos destacados pelo protótipo apresentado é a sua capacidade de fornecer *insights* com base nas entradas manuais. Entretanto, como parte de trabalhos futuros, planeja-se automatizar essas entradas por meio da mineração de logs dos sistemas de redes de computadores. Essa abordagem não apenas economizaria o tempo precioso dos administradores, mas também habilitaria respostas mais ágeis e eficientes a eventos e problemas na rede.

Além disso, a proposta de incluir uma saída numérica em arquivos de configuração de aplicações, que poderiam limitar automaticamente a largura de banda de um usuário, é relevante na gestão proativa da qualidade de serviço (QoS). Isso além de assegurar uma distribuição mais justa dos recursos de rede, também auxilia na prevenção de congestionamentos e na manutenção eficaz do funcionamento da rede.

Em retrospectiva, este trabalho se enquadra na abordagem da literatura sobre Redes Definidas por Software (SDN). Ao oferecer um conteúdo que desbrava as complexidades desse campo, foi possível proporcionar aos novos pesquisadores e profissionais um recurso que os auxilia na compreensão das nuances teóricas e práticas das SDN. Durante esta jornada de pesquisa e desenvolvimento, destacou-se os fundamentos, arquitetura, controladores, programação, orquestração, e casos de uso, proporcionando aos leitores um panorama dessa tecnologia. Também mostrou-se os caminhos mais simples para a prática da SDN utilizando os *frameworks* já disponíveis.

Ademais, ao abordar os conceitos atuais e as tendências futuras, é possível inferir em novos pesquisadores a respeito do futuro das Redes Definidas por Software. Este trabalho deve inspirar e capacitar novos pesquisadores a explorar ainda mais esse domínio, contribuindo assim para o avanço contínuo da pesquisa e da prática em SDN, e para a criação de redes mais flexíveis, escaláveis e eficientes para as gerações futuras.

Este trabalho exemplificou a aplicação da Lógica Fuzzy em conjunto com a linguagem de programação Python para a construção de uma solução destinada à determinação da largura de banda em redes de computadores. Destaca-se a relevância crítica e a viabilidade intrínseca de um sistema de gerenciamento de largura de banda eficaz em contextos de SDN.

À medida que avançamos, contemplamos um horizonte repleto de oportunidades e desafios, com a expectativa de que futuros trabalhos poderão documentar a utilização de outras funções de pertinência neste protótipo e implementar esse arcabouço em conjunto com a plataforma Mininet, transformando-o em um produto de software experimental plenamente funcional.

A trajetória delineada por este trabalho é um testemunho do potencial das Redes Definidas por Software (SDN). Conforme a comunidade científica se aproxima do desenvolvi-

mento de um produto de software que aprimorará o gerenciamento de banda em SDN, é antecipada uma contribuição para a eficiência e a escalabilidade das redes modernas.

As pesquisas acadêmicas dedicadas às Redes Definidas por Software mantêm sua vitalidade e relevância na academia, impulsionadas por uma ambição de expandir as capacidades desse sistema. A contínua exploração desse campo reflete a convicção de que as SDNs têm o potencial de transcender as redes tradicionais e se tornar uma infraestrutura fundamental para os Sistemas Autônomos da Internet. À medida que os desafios técnicos e conceituais são superados, os acadêmicos buscam a realização desse ambicioso objetivo.

Além disso, a penetração das soluções corporativas de Redes Definidas por Software, como as redes IBN da Cisco, os diversos produtos da VMWare e os esforços da Ubiquiti no ambiente empresarial, indica claramente que a tecnologia SDN está ganhando tração no mercado. A integração de SDN com Inteligência Artificial representa um passo significativo em direção à automação avançada e ao gerenciamento de redes de forma mais eficiente e adaptativa. Essas implementações empresariais demonstram que a SDN não é apenas uma ideia acadêmica abstrata, mas uma tecnologia concreta com aplicações práticas e impacto tangível no setor de comunicações.

Em suma, os softwares de auxílio para administradores de redes desempenham um papel essencial na simplificação da gestão de redes de computadores. À medida que essas ferramentas evoluem, com a automação de entradas e a implementação de políticas de QoS mais sofisticadas, elas contribuem para a melhoria da estabilidade, desempenho e confiabilidade das redes, atendendo às crescentes demandas de um mundo cada vez mais conectado.

Os resultados obtidos a partir desses estudos têm o potencial de contribuir para o campo de gerenciamento de largura de banda em redes de computadores. Eles podem lançar luz sobre a viabilidade de incorporar a Lógica Fuzzy como parte integrante de sistemas de controle de largura de banda, com implicações práticas para otimização de recursos, melhoria da qualidade de serviço (QoS) e satisfação do usuário.

Assim, é possível vislumbrar um horizonte para o futuro das Redes Definidas por Software (SDN). As pesquisas acadêmicas e as iniciativas empresariais estão colaborando de maneira harmoniosa para desvelar o potencial dessa tecnologia. Esse alinhamento nos leva a antever um cenário no qual as SDNs têm o potencial de substituir, em escala global, as infraestruturas de redes tradicionais.

REFERÊNCIAS

AGUADO, A. G.; CANTANHEDE, M. A. Lógica fuzzy. *Faculdade de Tecnologia – Universidade Estadual de Campinas*, 2010. Disponível em: <<https://docplayer.com.br/32988870-Logica-fuzzy-alexandre-garcia-aguado-marco-andre-cantanhede.html>>. Acesso em: 28/11/2021. Citado 2 vezes nas páginas 42 e 45.

ARBIZA, L. M. R. *SDN no Contexto de IoT: Refatoração de Middleware para Monitoramento de Pacientes Crônicos Baseada em Software-Defined Networking*. Tese (Mestrado em Ciência da Computação) — Pós-Grauação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Belém, 2016. Citado na página 51.

ARSLAN, A.; KAYA, M. Determination of fuzzy logic membership functions using genetic algorithms. *Fuzzy Sets and Systems*, v. 118, n. 2, p. 297–306, 2001. ISSN 0165-0114. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0165011499000652>>. Acesso em: Acessado em: junho de 2023. Citado na página 48.

ASHOORI, A. *QoS Management Solution in Software Defined Networking using Ryu Controller*. 2020. Disponível em: <https://github.com/amirashoori7/sdn_qos>. Acesso em: Acessado em: Fevereiro de 2023. Citado na página 63.

ASSIS, S. S. d.; ERRICO, L. d. *Uma abordagem multiobjetivo para definição de rotas em Redes Definidas por Software (SDN)*. 2014. Disponível em: <https://www.researchgate.net/profile/Luciano-Errico/publication/349856833_Uma_abordagem_multiobjetivo_para_definicao_de_rotas_em_Redes_Definidas_por_Software_SDN/links/606c6cf0a6fdccf289fd348d/Uma-abordagem-multiobjetivo-para-definicao-de-rotas-em-Redes-Definidas-por-Software-SDN.pdf>. Citado na página 53.

BARBOZA, L. M. *Aprovisionamento de banda sob demanda usando redes definidas por software*. Dissertação (Mestrado) — Universidade Federal Fluminense, 2019. Disponível em: <<https://app.uff.br/riuff/handle/1/12539>>. Citado 2 vezes nas páginas 22 e 23.

BARELLA, B. P. *Modelagem com Aprendizado de Máquina Aplicada aos Sistemas de Monitoramento de Integridade Estrutural*. Tese (Mestrado em Modelagem e Otimização) — Universidade Federal de Catalão, Catalão, Goiás, 2020. Citado 2 vezes nas páginas 32 e 33.

BHATTACHARYYA, R.; MUKHERJEE, S. Fuzzy membership function evaluation by non-linear regression: An algorithmic approach. *Fuzzy Information and Engineering*, Taylor and Francis, v. 12, n. 4, p. 412–434, 2020. Citado na página 49.

BLENK, A. *et al.* Control plane latency with sdn network hypervisors: the cost of virtualization. *IEEE Transactions on Network and Service Management*, v. 13, n. 3, p. 366–380, 2016. Citado na página 21.

BRETÃO, N. 74% das transações bancárias em abril foram feitas pelos canais digitais. 2020. Cisco.com. Disponível em: <<https://valorinveste.globo.com/produtos/servicos-financeiros/noticia/2020/06/23/74percent-das-transacoes-bancarias-em-abril-foram-feitas-pelos-canais-digitais.ghml>>. Acesso em: 29/10/2021. Citado na página 19.

BRITO, S. Com queda do Facebook, Mark Zuckerberg perde U\$ 6 bilhões em horas. 2021. Revista Veja. Disponível em: <<https://veja.abril.com.br/tecnologia/com-queda-do-facebook-mark-zuckerberg-perde-u-6-bilhoes-em-horas/>>. Acesso em: 14/10/2021. Citado na página 19.

BULHÕES, G. Facebook explica motivo para a queda de aplicativos por seis horas. 2021. Olhar Digital. Disponível em: <<https://olhardigital.com.br/2021/10/05/internet-e-redes-sociais/facebook-explica-motivo-para-a-queda-de-aplicativos-por-seis-horas/porGabrielaBulhÃmes>>. Acesso em: 14/10/2021. Citado na página 19.

CAMPOS, M. B.; MARTINS, J. O equivalente em carbonato de cálcio dos corretivos da acidez dos solos. *Revista Brasileira de Computação Aplicada* (ISSN 2176-6649, Passo Fundo, v. 9, n. 1, p. 107–119, 2017. Citado na página 20.

CENTENO, A. G. *et al.* Controladores sdn, elementos para su selección y evaluación. *Temática*, v. 13, n. 3, p. 10–20, 2021. Acesso em: 11/11/2021. Citado na página 51.

CHOLLET, F.; FRANCOIS, C. *Deep Learning With Python*. [S.l.]: Manning Publications Co., 2018. Citado 2 vezes nas páginas 32 e 33.

CISCO. *Redes Definidas Por Software*. 2021. Cisco.com. Disponível em: <https://www.cisco.com/c/pt_br/solutions/software-defined-networking/overview.html>. Acesso em: 20/09/2021. Citado 3 vezes nas páginas 15, 16 e 20.

_____. *CISCO NETWORKING AT-a-Glance*. 2023. Disponível em: <<https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/nb-06-networking-aag-cte-en.html>>. Citado 2 vezes nas páginas 55 e 56.

COX, E. The fuzzy systems handbook: a practitioner's guide to building, using, and maintaining fuzzy systems. *New York: AP Professional*, 1994. Acesso em: 28/11/2021. Citado na página 45.

FARES, A. A. Y. R. *Proposta de integração de um sistema de detecção de intrusão (IDS) entre uma rede SDN e uma honeynet*. 2021. Disponível em: <<https://repositorio.unb.br/handle/10482/41908>>. Acesso em: Acesso em 28 de junho de 2022. Citado 3 vezes nas páginas 25, 30 e 31.

FARIAS, F. N. *et al.* vSDNBox: Um hardware especializado de baixo custo gerenciado via SDN. In: SBC. *Anais do XXIII Workshop de Gerência e Operação de Redes e Serviços*. 2018. Disponível em: <<https://sol.sbc.org.br/index.php/wgrs/article/view/2687>>. Acesso em: Acesso em 10 de janeiro de 2022. Citado na página 29.

FAYAZ, M.; ULLAH, I.; KIM, D. An optimized fuzzy logic control model based on a strategy for the learning of membership functions in an indoor environment. *Electronics*, v. 8, n. 2, 2019. ISSN 2079-9292. Disponível em: <<https://www.mdpi.com/2079-9292/8/2/132>>. Acesso em: Acessado em: junho de 2023. Citado na página 49.

FERNANDES, A. F. F. C.; CARMO, F. d. A. S. D.; FERNANDES, R. R. SDN-redes definidas por software—por quê? *ResearchGate*, 2015. Disponível em: <https://www.researchgate.net/profile/Francisco-Carmo/publication/284190288_SDN_-_Software_Defined_Network_-_Por_que/links/564fb50f08aeafc2aab3fc23/SDN-Software-Defined-Network-Por-que.pdf>. Acesso em: Acesso em: 30 de junho de 2022. Citado na página 25.

FERNEDA, E.; DIAS, G. A. A lógica fuzzy aplicada a recuperação de informação. *InterScientia, João Pessoa*, p. 51–65, 2013. Disponível em: <<https://www.marilia.unesp.br/Home/Instituicao/Docentes/EdbertoFerneda/artigo-completo.pdf>>. Acesso em: 12/11/2021. Citado na página 37.

GALICIA, C. T. de Telecomunicacions de. *OpenFlow QoS Proof of Concept*. 2020. Disponível em: <<https://github.com/Gradiant/SDN-QoS-PoC>>. Acesso em: Acessado em: Fevereiro de 2023. Citado na página 64.

GERON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. [S.l.]: O'Reilly Media, 2019. Citado na página 33.

GOMES, V. S. L. S. *Isolamento de Recursos em Redes Definidas Por Software Virtualizadas Baseadas em Openflow*. Tese (Doutorado) — Universidade Federal do Pará, 2013. Disponível em: <<https://www.ppgcc.propesp.ufpa.br/index.php/br/teses-e-dissertacoes/dissertacoes/161-2013>>. Acesso em: Acesso em: 16 de Setembro de 2022. Citado na página 27.

GONDIM, E. B. *Monitoramento de desempenho com middleboxes em redes definidas por software*. Tese (Doutorado) — Universidade de Brasília, 2015. Disponível em: <<https://repositorio.unb.br/handle/10482/19792>>. Acesso em: Acesso em: 22 de julho de 2022. Citado na página 30.

GONZALEZ, J. *Real-time QoS and Call admission Control in SDN networks*. 2020. Disponível em: <<https://github.com/joagonzalez/sdn-qos>>. Acesso em: Acessado em: Fevereiro de 2023. Citado na página 63.

GUEDES, D. *et al.* Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. In: *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*. [s.n.], 2012. v. 30, n. 4, p. 160–210. Disponível em: <https://www.researchgate.net/publication/260346033_Redes_Definidas_por_Software_uma_abordagem_sistemica_para_o_desenvolvimento_de_pesquisas_em_Redes_de_Computadores>. Acesso em: Acesso em 09 de agosto de 2022. Citado 2 vezes nas páginas 21 e 22.

GUTZ tephen *et al.* Splendid isolation: A slice abstraction for software-defined networks. *HotSDN '12 - ACM*, 2012. Disponível em: <<https://dl.acm.org/doi/abs/10.1145/2342441.2342458>>. Acesso em: 10/11/2021. Citado na página 52.

HASAN, F.; SOBHAN, A. Describing fuzzy membership function and detecting the outlier by using five number summary of data. *American Journal of Computational Mathematics*, p. 410–424, 2020. Citado na página 48.

ITU-T. *ITU-T E.800: Quality of telecommunication services: concepts, models, objectives and dependability planning – Terms and definitions related to the quality of telecommunication services*. [S.l.], 2008. Citado na página 60.

JAIN, R.; PAUL, S. Network virtualization and software defined networking for cloud computing: a survey. *Communications Magazine - IEEE*, USA, v. 51, n. 11, p. 24–31, 2013. Citado na página 21.

JÚNIOR, E. P. F.; TAVARES, A. C. J.; NOGUEIRA, M. Lógica paraconsistente anotada aplicada na detecção de ataques ddos em redes sdn. *UFPR /UTFPR, RNP/GT-Periscope, SBC*, 2019. Disponível em: <<https://sol.sbc.org.br/index.php/sbseg/article/view/13995>>. Acesso em: 17/11/2021. Citado na página 51.

KELLER, E.; REXFORD, J. The "platform as a service" model for networking. *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, USA, v. 4, n. 5, 2010. Citado na página 21.

LEMOES, A.; PASTOR, L.; OLIVEIRA, N. Wi-fi salvador: mapeamento colaborativo e redes sem fio no brasil. *Intercom: Revista Brasileira de Ciências da Comunicação*, 2012. Disponível em: <<https://www.scielo.br/j/intercom/a/ZdyBpkFPw7DCvDhFpkBygZk/?lang=pt>>. Acesso em: 17/11/2021. Citado na página 14.

LEVIN, D. *et al.* Logically centralized?: state distribution trade-offs in software defined networks. Helsinki, Finlandia: [s.n.], 2012. 1-6 p. HotSDN '12 - ACM. Disponível em: <<https://dl.acm.org/doi/10.1145/2342441.2342443>>. Acesso em: 15/10/2021. Citado 2 vezes nas páginas 21 e 50.

MARTINS, E. Lógica booleana? saiba um pouco mais sobre esta lógica e como ela funciona. *Tecmundo.com.br*, 2009. Disponível em: <<https://www.tecmundo.com.br/programacao/1527-logica-booleana-saiba-um-pouco-mais-sobre-esta-logica-e-como-ela-funciona.htm>>. Acesso em: 11/11/2021. Citado na página 34.

MCKEWON, N. *et al.* Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun.*, v. 38, n. 2, p. 69–74, 2008. Disponível em: <<https://dl.acm.org/doi/10.1145/1355734.1355746>>. Acesso em: 21/10/2021. Citado na página 28.

MENOSSEI, J. *Avaliação de desempenho: Virtualizadores de rede*. 2015. Disponível em: <https://www.fibre.org.br/wp-content/uploads/2015/10/RF-extensao-Relatorio_Final-JeanMenossi.pdf>. Acesso em: 2022-07-15. Citado na página 26.

MININET. *Mininet: An Instant Virtual Network on your Laptop (or other PC)*. 2023. Mininet.org. Disponível em: <<https://mininet.org/re.com/br/topics/glossary/content/software-defined-networking.html>>. Acesso em: Acessado em: julho de 2023. Citado 2 vezes nas páginas 31 e 32.

MOLINOS, D. N. *et al.* NEA: arquitetura de elemento de rede SDN com suporte a MAC definido pela aplicação. 2022. Disponível em: <<http://repositorio.ufu.br/handle/123456789/35087>>. Acesso em: 2022-07-17. Citado na página 22.

MOURA, H. D. *Ethanol: uma plataforma SDN para redes wi-fi*. Dissertação (Mestrado) — Universidade federal de Minas Gerais, 2015. Disponível em: <<https://repositorio.ufmg.br/handle/1843/ESBF-9ZPNXT>>. Citado na página 30.

NETO, F. J. B. V.; MIGUEL, C. J.; SAMPAIO, P. N. M. Reprogramação de fluxos de dados em redes sdn baseada na análise contextual do tráfego. *SBC - Anais do XXVI Workshop de Gerência e Operação de Redes e Serviços*, v. 1, n. 1, 2021. Disponível em: <<https://sol.sbc.org.br/index.php/wgrs/article/view/17190>>. Acesso em: 11/11/2021. Citado 2 vezes nas páginas 51 e 52.

OLIVEIRA, S.; ALMEIDA, P. E. M. de; EVSUKOFF, A. G. *Sistemas Inteligentes: fundamentos e aplicações*. [S.l.]: Manole, São Paulo, 2003. Citado 2 vezes nas páginas 43 e 44.

OPEN NETWORKING FOUNDATION. *Software-Defined Networking: The new norm for networks*. Palo Alto, CA, USA, 2012. Citado na página 20.

PACÍFICO, R. D. *et al.* Roteador sdn em hardware independente de protocolo com análise, casamento e ações dinâmicas. In: SBC. *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. 2018. p. 407–420. Disponível em: <<https://sol.sbc.org.br/index.php/sbrc/article/view/2431>>. Acesso em: 02 de dezembro de 2022. Citado na página 54.

PELALAK, R. *et al.* Influence of machine learning membership functions and degree of membership function on each input parameter for simulation of reactors. *Scientific Reports*, v. 11, n. 1, 2021. Citado na página 49.

PINHEIRO, B. *et al.* Cim-sdn: A common information model extension for software-defined networking. *Globecom Workshops (GC Wkshps) - IEEE*, p. 836–841, 2013. Citado 2 vezes nas páginas 23 e 24.

PINHEIRO, B. A. *Uma Abordagem SDN para Virtualização de Redes*. Tese (Doutorado em Engenharia Elétrica) — Pós-Grauação em Engenharia Elétrica, Universidade Federal do Pará, Belém, 2016. Citado 6 vezes nas páginas 15, 21, 24, 26, 27 e 28.

PRAJAPATI, S.; FERNANDEZ, E. Performance evaluation of membership function on fuzzy logic model for solar pv array. *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, p. 609–613, 2020. Citado 2 vezes nas páginas 48 e 49.

REZENDE, P. *et al.* Roteamento multicaminhos em redes definidas por software. In: *2016 XXXIV Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*. [s.n.], 2016. p. 528–541. Disponível em: <<http://www.sbrc2016.ufba.br/downloads/SessoesTecnicas/152479.pdf>>. Acesso em: 20 de janeiro de 2023. Citado 2 vezes nas páginas 54 e 55.

RIBEIRO JÚNIOR, J.; GOMES JÚNIOR, C. S.; MOURA JÚNIOR, J. d. R. V. d. Lógica fuzzy aplicada na definição da largura de banda máxima de um usuário de rede de computadores. *Revista Mediação - UEG Pires do Rio*, v. 17, n. 1, p. 129–151, 2022. Citado 17 vezes nas páginas 34, 35, 38, 39, 40, 41, 42, 44, 65, 66, 67, 68, 70, 71, 72, 73 e 74.

RIGNEL, D. G. d. S.; CHENCI, G. P.; LUCAS, C. A. Uma introdução a lógica fuzzy. *Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica*, v. 01, n. 01, 2011. Disponível em: <http://www.logicafuzzy.com.br/wp-content/uploads/2013/04/uma_introducao_a_logica_fuzzy.pdf>. Acesso em: Acessado em: junho de 2023. Citado na página 34.

RODRIGUES, T. N.; GUIMARÃES, R. P. Usabilidade de controladores openflow: uma proposta de avaliação baseada em arquiteturas de código-fonte. *Abakós*, v. 3, n. 1, p. 38–58, 2014. Disponível em: <<http://periodicos.pucminas.br/index.php/abakos/article/view/8283>>. Acesso em: 13 de junho de 2022. Citado na página 28.

ROSS, T. J. *Fuzzy Logic with Engineering Applications*. Reino Unido: John Wiley & Sons Ltd, 2010. ISBN 978-0-470-74376-8. Citado na página 38.

ROTHENBERG, C. E. *et al.* Openflow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Cad. CPqD Tecnologia*, v. 7, n. 1, p. 65–76, 2010. Disponível em: <<https://www.academia.edu/download/34647347/artigo6.pdf>>. Acesso em: 14 de julho de 2022. Citado na página 29.

SALOMÃO, M. *Apagão na área de TI: sobram vagas, mas falta mão de obra*. 2021. Correio Brasiliense. Disponível em: <<https://www.correiobraziliense.com.br/euestudante/trabalho-e-formacao/2021/05/4926392-apagao-na-area-de-ti-sobram-vagas-mas-falta-mao-de-obra.html>>. Acesso em: 14/10/2021. Citado 2 vezes nas páginas 15 e 19.

SAMPAIO, L. M. D.; OLIVEIRA, M. J. F. de; IGNACIO, A. A. V. Lógica nebulosa: Aplicações e tendências. *SPOLM2007*, ISSN 2175-6295, 2007, 2007. Disponível em: <<https://www.marinha.mil.br/spolm/sites/www.marinha.mil.br/spolm/files/033.pdf>>. Acesso em: 11/11/2021. Citado 2 vezes nas páginas 36 e 37.

SANDRI, S.; CORREA, C. Lógica nebulosa. *INPE, CNRN*, 1999. Disponível em: <<http://www.ele.ita.br/cnrn/minicursos-5ern/log-neb.pdf>>. Acesso em: 12/11/2021. Citado 5 vezes nas páginas 36, 37, 38, 40 e 65.

SCHWEITZER, C. M.; FERRARI, R. C. C.; SHINODA, A. A. Análise de tráfego entre ovs e controlador sdn. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 6, n. 1, 2018. Disponível em: <<https://proceedings.sbmec.org.br/sbmec/article/view/2090>>. Acesso em: 30 de junho de 2022. Citado na página 25.

SHERWOOD, R. *et al.* Carving research slices out of your production networks with openflow. *SIGCOMM Comput. Commun.*, v. 40, n. 1, p. 129–130, 2010. Disponível em: <<https://dl.acm.org/doi/10.1145/1672308.1672333>>. Acesso em: 21/10/2021. Citado 2 vezes nas páginas 26 e 27.

_____. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium*, p. 1–13, 2009. Citado 2 vezes nas páginas 26 e 27.

SILVA, D. d. P. *et al.* Uma arquitetura para o provisionamento de qos interdomínios em redes virtuais baseadas no openflow. In: *31º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2013. Disponível em: <<http://sbrc2013.unb.br/files/anais/trilha-principal/artigos/artigo-62.pdf>>. Acesso em: julho de 2022. Citado na página 29.

SILVA, L. S. d. *ALGORITMO PARA BALANCEAMENTO DE CARGA DINÂMICO EM SDN*. Master's Thesis — Pontifícia Universidade Católica de Minas Gerais, 2018. Disponível em: <http://www.biblioteca.pucminas.br/teses/Informatica_SilvaLS_1.pdf>. Acesso em: 19 de janeiro de 2022. Citado na página 53.

- SOUZA, G. C. d. Trabalho de Conclusão de Curso, *Implementação de um controlador ODL para SDN*. 2019. Disponível em: <<http://repositorio.utfpr.edu.br/jspui/handle/1/23950>>. Acesso em: 2022-06-30. Citado 2 vezes nas páginas 22 e 30.
- SOUZA, G. M. d. *Um mecanismo inteligente de roteamento para Redes Definidas por Software baseadas no protocolo OpenFlow*. 2014. Disponível em: <<https://www.rii.ufam.edu.br/handle/prefix/3762>>. Acesso em: 2022-12-04. Citado na página 54.
- SPALLA, E. S. *Estratégias para resiliência em SDN: Uma abordagem centrada em multi-controladores ativamente replicados*. 2015. Disponível em: <<http://repositorio.ufes.br:8080/handle/10/4287>>. Acesso em: 2022-08-02. Citado na página 29.
- TANENBAUM, A. S.; WETHERALL, D. J. *Redes de Computadores*. São Paulo: Bookman, 2021. Citado 3 vezes nas páginas 20, 60 e 66.
- TANSCHKEIT, R. *Sistemas Fuzzy*. 2005. Notas de Aula DEE-PUC-Rio. Disponível em: <<https://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/LN-Sistemas%20Fuzzy.pdf>>. Acesso em: Acessado em: junho de 2023. Citado na página 39.
- TEIXEIRA, E. V. *Virtualização (SDN e NFV) com ênfase em desempenho de rede*. 2018. Disponível em: <<http://rii.utfpr.edu.br/jspui/handle/1/19367>>. Acesso em: 2022-06-29. Citado na página 25.
- TELECOMUNICAÇÕES, M. das. *Internet das Coisas: um passeio pelo futuro que já é realidade no dia a dia das pessoas*. 2021. Ministério das Telecomunicações. Disponível em: <<https://www.gov.br/mcom/pt-br/noticias/2021/marco/internet-das-coisas-um-passeio-pelo-futuro-que-ja-e-real-no-dia-a-dia-das-pessoas>>. Acesso em: 14/10/2021. Citado 2 vezes nas páginas 14 e 20.
- TOCCI, R. J.; WILMER, N.; MOSS, G. *Sistemas Digitais: princípios e aplicações*. São Paulo: Pearson, 2011. Citado na página 34.
- TSAI, P.-W.; RISDIANTO, A. C.; LING, T. C. Design, development, and operation of a sdn-based bgp playground for networkers. In: *2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS)*. [S.l.: s.n.], 2022. p. 1–4. Citado na página 54.
- TURING, A. Computing machinery and intelligence. *Mind*, v. 59, n. 236, p. 433–460, 1950. Acesso em: 05/11/2021. Citado na página 32.
- UBIQUITI. *Arquitetura SDN e Solução Enterprise UniFi*. 2018. Medium.com. Disponível em: <<https://medium.com/ubntbr/arquitetura-sdn-e-solu%C3%A7%C3%A3o-enterprise-unifi-ba44a58d92c9>>. Acesso em: Acessado em: julho de 2023. Citado 3 vezes nas páginas 15, 56 e 57.
- VAIDYA, A. V.; METKEWAR, P. S.; NAIK, S. A. A new paradigm for generation of fuzzy membership function. *2018 IEEE 8th International Advance Computing Conference (IACC)*, p. 1–6, 2018. Citado na página 50.
- VMWARE. *Redes definidas por software - VMWare*. 2023. VMWare. Disponível em: <<https://www.vmware.com/br/topics/glossary/content/software-defined-networking.html>>. Acesso em: Acessado em: julho de 2023. Citado 2 vezes nas páginas 58 e 59.

VOLPATO, F. *et al.* *Arquitetura para o gerenciamento autônomo de QOS em ambientes SDN*. 2017. Disponível em: <<https://repositorio.ufsc.br/handle/123456789/188082>>. Citado na página 21.

YILDIZ, Z. C. *A Short Fuzzy Logic Tutorial*. Bilkent University, 2010. Disponível em: <http://cs.bilkent.edu.tr/~zeynep/files/short_fuzzy_logic_tutorial.pdf>. Acesso em: 13/11/2021. Citado 5 vezes nas páginas 40, 41, 43, 44 e 45.

APÊNDICE A

Código Colab

No trecho de código 3.1 tem-se uma implementação do sistema.

Código A.1 – Instalar o Pacote

```
1 # ETAPA 0: INSTALAR O PACOTE FUZZY
2 !pip install -U scikit-fuzzy
```

Código A.2 – Importar as Bibliotecas

```
1 # ETAPA 1: IMPORTAR AS BIBLIOTECAS QUE SERAO USADAS (FUZZY, VETORES E
  GRAFICOS)
2 import numpy as np
3 import skfuzzy as fuzz
4 import matplotlib.pyplot as plt
```

Código A.3 – Conjuntos Fuzzy

```
1 ETAPA 2: CRIANDO OS CONJUNTOS FUZZY DE SUAS VARIAVEIS
2 # Construindo os vetores das variaveis
3 # * Categoria de prioridade de servico [0, 100]
4 # * Classificacao da necessidade do usuario [0, 100]
5 # * numero de usuarios dividindo a rede[0 - 1000] -> Considerando uma
  empresa com ate 1000 funcionarios
6 #Saida
7 # * Porcentagem concedida da banda ao usuario [0, 100]
8 x_categoria_servico = np.arange(-1.0, 101, 1)
9 x_necesUsr = np.arange(-1.0, 101, 1)
10 x_nusuario = np.arange(-1.0, 1001, 1)
11
12 x_velocidade = np.arange(-1.0, 101, 1)
```

Código A.4 – Criando as Funções de Pertinência

```

1
2 # Criando as FUNCOES DE PERTINENCIA FUZZY
3 # AS FUNCOES PODEM TER FORMAS DIFERENTES:
4 # trimf(x,[a,b,c]) - 3 pontos p definir o triangulo
5 # trapmf(x, [a,b,c,d]) - 4 pontos p definir o trapezio
6 # smf(x,a,b) - a: onde a fc comeca a subir de zero; b: qdo a fc alcanca
7   1
8 # sigmf(x,b,c) - sigmoide - b: centro da curva (vies); c: largura da
9   curva
10 # gaussmf(x,mean,sigma) - gaussiana
11 # gbellmf(x, largura, centro, declive) - fc bell generalizada  $y(x) = 1 / (1$ 
12    $+ \text{abs}([x - c] / a) ** [2 * b])$ 
13 # piecemf(x,[a,b,c]) - fc por partes
14 # pimf(x,a,b,c,d) - fc pi
15
16 categoria_servico_lo = fuzz.trimf(x_categoria_servico, [0, 0, 40])
17 categoria_servico_lo2 = fuzz.trimf(x_categoria_servico, [0, 40, 60]) #20
18 categoria_servico_md = fuzz.trimf(x_categoria_servico, [40, 60, 80])
19 categoria_servico_md2 = fuzz.trimf(x_categoria_servico, [60, 80, 100])
20 categoria_servico_hi = fuzz.trimf(x_categoria_servico, [80, 100, 100])
21
22 necesUsr_lo = fuzz.trimf(x_necesUsr, [0, 0, 50])
23 necesUsr_lo2 = fuzz.trimf(x_necesUsr, [0, 50, 65]) #25
24 necesUsr_md = fuzz.trimf(x_necesUsr, [50, 65, 80])
25 necesUsr_md2 = fuzz.trimf(x_necesUsr, [80, 90, 100])
26 necesUsr_hi = fuzz.trimf(x_necesUsr, [90, 100, 100])
27
28 nusuario_lo = fuzz.trimf(x_nusuario, [0, 0, 500])
29 nusuario_lo2 = fuzz.trimf(x_nusuario, [0, 500, 650]) #300
30 nusuario_md = fuzz.trimf(x_nusuario, [500, 650, 850])
31 nusuario_md2 = fuzz.trimf(x_nusuario, [650, 850, 1000])
32 nusuario_hi = fuzz.trimf(x_nusuario, [850, 1000, 1000])
33
34 velocidade_lo = fuzz.trimf(x_velocidade, [0, 0, 40])
35 velocidade_lo2 = fuzz.trimf(x_velocidade, [0, 40, 60]) #20
36 velocidade_md = fuzz.trimf(x_velocidade, [40, 60, 80])
37 velocidade_md2 = fuzz.trimf(x_velocidade, [60, 80, 100])
38 velocidade_hi = fuzz.trimf(x_velocidade, [80, 100, 100])

```

Código A.5 – Visualizando as funções de Pertinência

```

1 #Visualizando as fcs de pertinencia
2 fig, (ax0, ax1, ax2, ax3) = plt.subplots(nrows=4, figsize=(8, 9))
3
4 ax0.plot(x_categoria_servico, categoria_servico_lo, 'b', linewidth=1.5,
5         label='Desnecessario')
6 ax0.plot(x_categoria_servico, categoria_servico_lo2, 'purple', linewidth=
7         =1.5, label='Pouco Necessario')
8 ax0.plot(x_categoria_servico, categoria_servico_md, 'y', linewidth=1.5,
9         label='Necessario')
10 ax0.plot(x_categoria_servico, categoria_servico_md2, 'g', linewidth=1.5,
11         label='Muito Necessario')
12 ax0.plot(x_categoria_servico, categoria_servico_hi, 'r', linewidth=1.5,
13         label='Indispensavel')
14 ax0.set_title('Categoria do Servico')
15 ax0.legend()
16
17 ax1.plot(x_necesUsr, necesUsr_lo, 'b', linewidth=1.5, label='Alta
18         necessidade')
19 ax1.plot(x_necesUsr, necesUsr_lo2, 'purple', linewidth=1.5, label='Alta
20         necessidade')
21 ax1.plot(x_necesUsr, necesUsr_md, 'y', linewidth=1.5, label='Media
22         necessidade')
23 ax1.plot(x_necesUsr, necesUsr_md2, 'g', linewidth=1.5, label='Media
24         necessidade')
25 ax1.plot(x_necesUsr, necesUsr_hi, 'r', linewidth=1.5, label='Baixa
26         necessidade')
27 ax1.set_title('necessidade do usuario')
28 ax1.legend()
29
30 ax2.plot(x_nusuario, nusuario_lo, 'b', linewidth=1.5, label='Baixa')
31 ax2.plot(x_nusuario, nusuario_lo2, 'purple', linewidth=1.5, label='Baixa
32         2')
33 ax2.plot(x_nusuario, nusuario_md, 'y', linewidth=1.5, label='Media')
34 ax2.plot(x_nusuario, nusuario_md2, 'g', linewidth=1.5, label='Media 2')
35 ax2.plot(x_nusuario, nusuario_hi, 'r', linewidth=1.5, label='Alta')
36 ax2.set_title('Densidade de Usuarios Conectados')
37 ax2.legend()
38
39 ax3.plot(x_velocidade, velocidade_lo, 'b', linewidth=1.5, label='Baixa')
40 ax3.plot(x_velocidade, velocidade_lo2, 'purple', linewidth=1.5, label='
41         Baixa 2')
42 ax3.plot(x_velocidade, velocidade_md, 'y', linewidth=1.5, label='Media')
43 ax3.plot(x_velocidade, velocidade_md2, 'g', linewidth=1.5, label='Media
44         2')
45 ax3.plot(x_velocidade, velocidade_hi, 'r', linewidth=1.5, label='Alta')
46 ax3.set_title('% de banda')
47 ax3.legend()
48
49 # Ocultando os eixos
50 for ax in (ax0, ax1, ax2):
51     ax.spines['top'].set_visible(False)
52     ax.spines['right'].set_visible(False)
53     ax.get_xaxis().tick_bottom()
54     ax.get_yaxis().tick_left()
55
56 plt.tight_layout()

```

Código A.6 – Fuzzyficando os Dados

```

1 # ETAPA 3: FUZZYFICANDO OS DADOS (CONVERTENDO SEUS VALORES REAIS EM
  FUZZY)
2 # Nesta etapa convertemos os dados de entrada do modelo, que sao
  numericos, em
3 # valores Fuzzy atraves do processo de fuzzyficacao
4 # A) x = Classificacao do protocolo / tipo de acesso http: site pronto
  ou p2p - 0.0%, stream de video com buffer(youtube) 30%, Redes sociais
  20%
5 # Streams ao vivo para video chamada - 70%, acesso aos servicos da
  empresa e sites comerciais 100%.
6 ## B) Qualidade prioridade de usuario de acordo com a necessidade: baixa
  , baixa-media, media, media-alta, alta.
7 ### C) A quantidade de usuarios conectados de 0 a 1000, o efeito e
  inverso, quanto mais usuarios pior.
8 # O modelo ira responder a pergunta: quantos % (0-100) de uso de banda
  de rede o usuario podera usar.
9 #Fazer a entrada manualmente qual o servico utilizado.
10 #Fazer a entrada manual da necessidade do usuario.
11
12 categoria_do_servico = 100.0
13 categoria_servico_level_lo = fuzz.interp_membership(x_categoria_servico,
  categoria_servico_lo, categoria_do_servico)
14 categoria_servico_level_lo2 = fuzz.interp_membership(x_categoria_servico
  , categoria_servico_lo2, categoria_do_servico)
15 categoria_servico_level_md = fuzz.interp_membership(x_categoria_servico,
  categoria_servico_md, categoria_do_servico)
16 categoria_servico_level_md2 = fuzz.interp_membership(x_categoria_servico
  , categoria_servico_md2, categoria_do_servico)
17 categoria_servico_level_hi = fuzz.interp_membership(x_categoria_servico,
  categoria_servico_hi, categoria_do_servico)
18
19 necessidade_do_usuario = 100.0
20 necesUsr_level_lo = fuzz.interp_membership(x_necesUsr, necesUsr_lo,
  necessidade_do_usuario)
21 necesUsr_level_lo2 = fuzz.interp_membership(x_necesUsr, necesUsr_lo2,
  necessidade_do_usuario)
22 necesUsr_level_md = fuzz.interp_membership(x_necesUsr, necesUsr_md,
  necessidade_do_usuario)
23 necesUsr_level_md2 = fuzz.interp_membership(x_necesUsr, necesUsr_md2,
  necessidade_do_usuario)
24 necesUsr_level_hi = fuzz.interp_membership(x_necesUsr, necesUsr_hi,
  necessidade_do_usuario)
25
26 quantidade_de_usuarios_conectados = 50.0
27 nusuario_level_lo = fuzz.interp_membership(x_nusuario, nusuario_lo,
  quantidade_de_usuarios_conectados)
28 nusuario_level_lo2 = fuzz.interp_membership(x_nusuario, nusuario_lo2,
  quantidade_de_usuarios_conectados)
29 nusuario_level_md = fuzz.interp_membership(x_nusuario, nusuario_md,
  quantidade_de_usuarios_conectados)
30 nusuario_level_md2 = fuzz.interp_membership(x_nusuario, nusuario_md2,
  quantidade_de_usuarios_conectados)
31 nusuario_level_hi = fuzz.interp_membership(x_nusuario, nusuario_hi,
  quantidade_de_usuarios_conectados)

```

Código A.7 – Vizualizando dados Fuzzyficados

```

1 print(categoria_servico_level_lo, categoria_servico_level_lo2,
      categoria_servico_level_md, categoria_servico_level_md2,
      categoria_servico_level_hi)
2 print(necesUsr_level_lo, necesUsr_level_lo2, necesUsr_level_md,
      necesUsr_level_md2, necesUsr_level_hi)
3 print(nusuario_level_lo, nusuario_level_lo2, nusuario_level_md,
      nusuario_level_md2, nusuario_level_hi )

```

Código A.8 – Aplicando Regras

```

1 # ETAPA 4: APLICANDO AS REGRAS EM SUA INSTANCIA (SEU CASO ESPECIFICO)
2 # Lembrando que aqui ainda estamos aplicando as regras para cada
   variavel,
3 # ainda NAO e a inferencia (juntando todas)
4
5 #Regra Velociade baixa
6 ativa_regra1 = np.fmax(categoria_servico_level_lo, np.fmax(
   necesUsr_level_lo, nusuario_level_hi)) #nusuario_HI siguinifica uma
   maior sobrecarga de usuarios.
7 ativavelocidade_lo = np.fmin(ativa_regra1, velocidade_lo)
8 #Regra Velociade baixa 2
9 ativa_regra2 = np.fmax(categoria_servico_level_lo2, np.fmax(
   necesUsr_level_lo2, nusuario_level_md2)) #nusuario_HI siguinifica
   uma maior sobrecarga de usuarios.
10 ativavelocidade_lo2 = np.fmin(ativa_regra2, velocidade_lo2)
11 #Regra Velocidade Media
12 ativa_regra3 = np.fmax(categoria_servico_level_md, np.fmax(
   necesUsr_level_md, nusuario_level_md))
13 ativavelocidade_md = np.fmin(ativa_regra3, velocidade_md)
14 #Regra Velociade Media 2
15 ativa_regra4 = np.fmax(categoria_servico_level_md2, np.fmax(
   necesUsr_level_md2, nusuario_level_lo2))
16 ativavelocidade_md2 = np.fmin(ativa_regra4, velocidade_md2)
17 #Regra Velociade Alta
18 ativa_regra5 = np.fmax(categoria_servico_level_hi, np.fmax(
   necesUsr_level_hi, nusuario_level_lo))
19 ativavelocidade_hi = np.fmin(ativa_regra5, velocidade_hi)
20
21 velocidade0 = np.zeros_like(x_velocidade)

```

Código A.9 – Visualizando Regras aplicadas

```

1 # Visualizando o conjunto de regras aplicadas
2 fig, ax0 = plt.subplots(figsize=(8, 3))
3
4 ax0.fill_between(x_velocidade, velocidade0, ativavelocidade_lo,
5                 facecolor='b', alpha=0.7)
6 ax0.plot(x_velocidade, velocidade_lo, 'b', linewidth=0.5, linestyle='--',
7         , )
8 ax0.fill_between(x_velocidade, velocidade0, ativavelocidade_lo2,
9                 facecolor='purple', alpha=0.7)
10 ax0.plot(x_velocidade, velocidade_lo2, 'purple', linewidth=0.5,
11         linestyle='--', )
12 ax0.fill_between(x_velocidade, velocidade0, ativavelocidade_md,
13                 facecolor='y', alpha=0.7)
14 ax0.plot(x_velocidade, velocidade_md, 'y', linewidth=0.5, linestyle='--',
15         )
16 ax0.fill_between(x_velocidade, velocidade0, ativavelocidade_md2,
17                 facecolor='g', alpha=0.7)
18 ax0.plot(x_velocidade, velocidade_md2, 'g', linewidth=0.5, linestyle='--',
19         ')
20 ax0.fill_between(x_velocidade, velocidade0, ativavelocidade_hi,
21                 facecolor='r', alpha=0.7)
22 ax0.plot(x_velocidade, velocidade_hi, 'r', linewidth=0.5, linestyle='--',
23         )
24 ax0.set_title('Saida Linguistica')
25
26 # Oculta eixos
27 for ax in (ax0,):
28     ax.spines['top'].set_visible(False)
29     ax.spines['right'].set_visible(False)
30     ax.get_xaxis().tick_bottom()
31     ax.get_yaxis().tick_left()
32
33 plt.tight_layout()

```

Código A.10 – Inferencia

```

1 # ETAPA 5: Inferencia = juntando todas as regras
2 inferencia = np.fmax(ativavelocidade_lo,
3                     np.fmax(ativavelocidade_lo2, np.fmax(
4                         ativavelocidade_md,
5                         np.fmax(ativavelocidade_md2,
6                             ativavelocidade_hi))))

```

Código A.11 – Defuzzificnado Dados

```

1 # ETAPA 6: Calculando o RESULTADO DEFUZZIFICADO
2 velocidadefinal = fuzz.defuzz(x_velocidade, inferencia, 'centroid')
3 ativavelocidade = fuzz.interp_membership(x_velocidade, inferencia,
    velocidadefinal) # plotando
4
5 # Visualizando o resultado
6 fig, ax0 = plt.subplots(figsize=(8, 3))
7
8 ax0.plot(x_velocidade, velocidade_lo, 'b', linewidth=0.5, linestyle='--',
    , )
9 ax0.plot(x_velocidade, velocidade_lo2, 'purple', linewidth=0.5,
    linestyle='--')
10 ax0.plot(x_velocidade, velocidade_md, 'y', linewidth=0.5, linestyle='--',
    )
11 ax0.plot(x_velocidade, velocidade_md2, 'g', linewidth=0.5, linestyle='--',
    ')
12 ax0.plot(x_velocidade, velocidade_hi, 'r', linewidth=0.5, linestyle='--',
    )
13 ax0.fill_between(x_velocidade, velocidade0, inferencia, facecolor='
    Orange', alpha=0.7)
14 ax0.plot([velocidadefinal, velocidadefinal], [0, ativavelocidade], 'k',
    linewidth=1.5, alpha=0.9)
15
16
17 ax0.set_title('Inferencia das Funcoes de Pertinencia e Resultado (LINHA)
    ')
18 # Ocultando os eixos x e y
19 for ax in (ax0,):
20     ax.spines['top'].set_visible(False)
21     ax.spines['right'].set_visible(False)
22     ax.get_xaxis().tick_bottom()
23     ax.get_yaxis().tick_left()
24
25 plt.tight_layout()
26
27 print("O usuario tera um total de" , velocidadefinal, "% da banda de
    download e upload disponivel como velocidade maxima de conexao nesta
    rede.")

```