Laboratório de Redes, Tecnologias e Serviços de Comunicação a Distância

Laboratório de Diagnóstico de Falha, Controle, Otimização e Modelagem

Departamento de Engenharia Eletrônica

Universidade Federal de Minas Gerais

Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG Brasil

Fone: +55 3499-4866 - Fax: +55 3499-4850

UFMG

UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Uma Abordagem Inovadora para Classificação de Tráfego em Tempo Real em Ambiente de Redes Definidas por Software (SDN)

Klenilmar Lopes Dias

Belo Horizonte - MG
2019

Laboratório de Redes, Tecnologias e Serviços de Comunicação a Distância

Laboratório de Diagnóstico de Falha, Controle, Otimização e Modelagem

Departamento de Engenharia Eletrônica

Universidade Federal de Minas Gerais

Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG Brasil

Fone: +55 3499-4866 - Fax: +55 3499-4850

UFMG

UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Uma Abordagem Inovadora para Classificação de Tráfego em Tempo Real em Ambiente de Redes Definidas por Software (SDN)

Klenilmar Lopes Dias

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientadores: Prof. Luciano de Errico, Ph.D.
Prof. Walmir Matos Caminhas, Dr.

Belo Horizonte - MG
2019

Dias, Klenilmar Lopes.
D541a Uma abordagem inovadora para classificação de tráfego em tempo real em ambiente de redes definidas por software (SDN) [recurso eletrônico] / Klenilmar Lopes Dias. - 2019.
1 recurso online (119 f. : il., color.) : pdf.

Orientador: Luciano de Errico.
Coorientador: Walmir Matos Caminhas.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 107-119.

Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia Elétrica - Teses. 2. Aprendizado do computador - Teses. 3. Teoria bayesiana de decisão estatística - Teses. 4. Redes de computadores – Teses. I. Errico, Luciano de. II. Caminhas, Walmir Matos. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Letícia Alves Vieira - CRB-6/2337
Biblioteca Prof. Mário Werneck - Escola de Engenharia da UFMG

"Uma Abordagem Inovadora para Classificação de Tráfego em Tempo Real em Ambiente de Redes Definidas por Software (SDN)"

Klenilmar Lopes Dias

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 02 de agosto de 2019.

Por:

Prof. Dr. Luciano de Errico
DELT (UFMG) - Orientador

Prof. Dr. Walmir Matos Caminhas
DELT (UFMG) - Coorientador

Prof. Dr. Eduardo Mazoni Andrade Marçal Mendes
DELT (UFMG)

Prof. Dr. Adriano Vilela Barbosa
DELT (UFMG)

Prof. Dr. Edmundo Roberto Mauro Madeira
IC (UNICAMP)

Prof. Dr. Joaquim Celestino Júnior
LARCES (UECE)

Dedicatória

A minha querida e amada esposa Michele Dias, que com sua fé em Deus, dedicação e seu amor incondicional durante todas as fases das nossas vidas, se tornou o alicerce desta tese.

Agradecimentos

A Deus por iluminar o meu caminho. Minha caminhada tem sido marcada por realizações diárias, que algumas vezes não dou o devido valor, mas sei que a graça de Deus se faz presente em todos os momentos da minha caminhada.

Devemos ser agradecidos a Deus, mas também às pessoas à nossa volta que nos abençoam. Com este sentimento expresso minha imensa gratidão aos meus orientadores Prof. Luciano de Errico e Prof. Walmir Matos Caminhas, por todos os direcionamentos, o empenho e a confiança que ajudaram a tornar possível este sonho tão especial. Vocês são exemplos de profissionais e seres humanos no seu maior grau de generosidade.

Expresso também minha gratidão ao Professor Reinaldo Martinez Palhares, pelo incentivo, pelas caronas no fim do dia, pelo café e pelas sugestões sempre oportunas e inteligentes.

Agradeço aos meus pais, José Ribamar Leal Dias e Francisca Lopes Dias, pelo amor e apoio incondicional durante todos os momentos de dificuldade que tive que enfrentar até chegar aqui. Também à todos meus irmãos, amo vocês!!

Agradeço de forma mais do que especial a minha grande e eterna esposa Michele Dias, pelo grande amor que tem por nossa família e por ter aceitado junto comigo essa empreitada de nossas vidas, abdicando de sua vida profissional pra cuidar da família. Aos meus quatro anjos da guarda, Lara, Felipe, Karla e Rebeka, que tiveram que conviver com minha ausência para que eu tivesse mais condições de educá-los, para que se tornem seres humanos dignos. Vocês foram a fonte inspiradora deste trabalho.

"É junto dos bão que a gente fica mió". É com essa aclamada frase de Guimarães Rosa que expresso meus agradecimentos aos dois laboratório LabCOM e D!FCOM e seus membros. Meus sinceros agradecimentos à todos os membros do LabCOM, especialmente ao Álvaro Lemos, Pedro Dias e Mateus Pongelupe, que vivenciaram comigo desde o inicio, compartilhando suas experiências e paciência. Ao amigo Pongelupe,

faço um adendo, por tudo isto e também pela parceria e amizade na fase do artigo, que até o final (Review 2) auxiliou por altas horas (via WhatsApp/E-mail) e com toda boa vontade aos questionamentos. Sou eternamente grato Pongelupe!!. Minha gratidão ao laboratório D!FCOM e ao seus integrantes, os quais eu tenho em grande estima. Aos amigos, em especial, Luciana Balieiro, Guilherme Costa (Guigui) e Artur Porto, que foram os primeiros a me receberem no laboratório e ao Wagner Alvarenga e Fúlvia Oliveira pelas opiniões, ajudas técnicas e companheirismo ao longo do doutorado.

Ao PPGEE e seus funcionários agradeço pelo suporte acadêmico. Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), agradeço pelo suporte financeiro.

Agradeço aos membros da banca examinadora deste trabalho pela avaliação do presente documento.

E por fim, gostaria de agradecer e lembrar da minha sogra Sueli (in **memoriam**), que sempre acompanhou meu crescimento profissional.

Epígrafe

"Ele vos deu vida, estando vós mortos nos vossos delitos e pecados.
Nos quais andastes outrora, segundo o curso deste mundo, segundo o príncipe da potestade do ar, do espírito que agora atua nos filhos da desobediência.
Entre os quais também todos nós andamos outrora, segundo as inclinações da nossa carne, fazendo a vontade da carne e dos pensamentos; e éramos, por natureza, filhos da ira, como também os demais.
Mas Deus, sendo rico em misericórdia, por causa do grande amor com que nos amou,
E estando nós mortos em nossos delitos, nos deu vida juntamente com Cristo - pela graça sois salvos,
E, juntamente com ele, nos ressuscitou, e nos fez assentar nos lugares celestiais em Cristo Jesus;
Para mostrar, nos séculos vindouros, a suprema riqueza da sua graça, em bondade para conosco, em Cristo Jesus.
Porque pela graça sois salvos, mediante a fé; e isto não vem de vós, é dom de Deus.
Não vem das obras, para que ninguém se glorie;
Pois somos feitura dele, criados em Cristo Jesus para boas obras, as quais Deus de antemão preparou para que andássemos nelas."

Ef 2:1-10

Resumo

A crescente demanda por taxas de transmissão de alta velocidade nos últimos anos atraiu pesquisas em novos mecanismos de caracterização e classificação de tráfego de rede. Seu tratamento inadequado degrada o desempenho de esquemas operacionais importantes, como Capacidade de Sobrevivência de Rede, Engenharia de Tráfego, Qualidade de Serviço (QoS) e Controle de Acesso Dinâmico, entre outros. Os métodos mais comuns para classificação de tráfego são a Inspeção Profunda de Pacotes (DPI) e a Classificação Baseada em Porta. No entanto, esses métodos estão se tornando obsoletos, já que cada vez mais tráfego está sendo criptografado e os aplicativos estão usando portas dinâmicas originalmente atribuídas a outros aplicativos populares. Nesta mesma direção, surge o problema de atender não só o volume futuro de tráfego nos Data Centers, mas atender também o tráfego atual e suas oscilações com essa nova demanda por taxas de transmissão de alta velocidade requisitada pelos serviços na Internet. Um dos principais problemas gerado pelo aumento do volume é em relação ao uso eficiente da taxa de transferência para atender as requisições. Além disso, nos últimos anos, o paradigma das Redes Definidas por Software (SDN) trouxe novas oportunidades para gerência da rede de maneira centralizada. Assim, este trabalho apresenta um classificador de tráfego para ambiente SDN, com uma abordagem inovadora de classificação de tráfego de rede em tempo real, para utilização em esquemas de Engenharia de Tráfego em Data Centers. O classificador adota uma nova abordagem para o relaxamento da hipótese de independência entre os atributos do algoritmo Naive Bayes. Os resultados mostram que o módulo proposto é uma alternativa promissora para uso em cenários em tempo real.

Palavras-chave: Classificação de Tráfego; Redes Definidas por Software; Aprendizado de Máquina; Naive Bayes; Streaming de Vídeo.

Abstract

The growing demand for high-speed transmission rates on recent years has attracted research in new mechanisms for network traffic characterization and classification. Their inadequate treatment degrades the performance of important operational schemes, such as Network Survivability, Traffic Engineering, Quality of Service (QoS), and Dynamic Access Control, among others. The most common methods for traffic classification are Deep Packet Inspection (DPI) and port based classification. However, those methods are becoming obsolete, as increasingly more traffic is being encrypted and applications are using dynamic ports or ports originally assigned to other popular applications. In this same direction, the problem arises not only in sustaining the future volume of traffic in Data Centers, but also in sustaining the current traffic and its oscillations with this new demand for high-speed transmission rates required by Internet services. One of the main problems generated by the increase in volume is the efficient use of the throughput, to attend the requisitions. In addition, in recent years the Software Defined Networks (SDN) paradigm has brought new opportunities for centralized network management. Thus, this work presents a SDN traffic classifier, with an innovative approach to real-time network traffic classification, for use in Traffic Engineering solutions in Data Centers. The classifier adopts a new approach for the relaxation of the hypothesis of independence between the attributes of the Naive Bayes algorithm. The results show that the proposed module is a promising alternative for use in real-time scenarios.

Keywords: Traffic Classification; Software Defined Networks; Machine Learning; Naive Bayes; Video Streaming.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AP	Algoritmo Proposto
API	Aplication Programming Interface
BoF	Bag-of-Flow
BPM	Beam Propagation Method
BR	Bayesian Regularization
BRPP	Backup Reprovisioning with Partial Protection
CBR	Constant Bit Rate
CLI	Command Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DFT	Discrete Fourier Transform
DNS	Domain Name System
DPI	Deep Packet Inspection
EGB	Extreme Gradiente Boosting
EON	Elastic Optical Network
FTP	File Transfer Protocol
FTTH	Fiber-To-The-Home
GA	Genetic Algorithm
GMPLS	Generalized Multi-Protocol Label Switching
GP	Genetic Programming
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LM	Levenberg-Marquardt
LTE	Long Term Evolution
Maxent	Maximum Entropy
ML	Machine Learning
MLP	Multilayer Perceptron
MPLS	MultiProtocol Label Switching
NARX	Nonlinear Autoregressive Model
NAT	Network Address Translation
NB	Naive Bayes
NBG	Naive Bayes Gaussiano
NN	Nearest Neighbor
ns-2	Network Simulator 2
OLT	Optical Line Terminal
PAI	Packet Arrival Interval
PCA	Principal Component Analysis
PD	Positive Definite
P2P	Peer-to-Peer
QoE	Quality of Experience
QoS	Quality of Service
RF	Random Forests
RTP	Real-time Transport Protocol
SGB	Stochastic Gradient Boosting

SDN	Software Defined Networks
SCG	Scaled Conjugate Gradient
SMTP	Simple Mail Transfer Protocol
SPD	Symmetric Positive Definite
SSH	Secure Shell
SSS	Spectrum Selective Switch
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TE	Traffic Enginnering
TI	Tecnologia da Informação
UDP	User Datagram Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WDM	Wavelength-Division Multiplex
VoIP	Voice over Internet Protocol

Sumário

Capítulo 1

Introdução

1.1 Motivação e Justificativa

Ao longo dos anos, tem havido um aumento do tráfego na Internet, com uma contribuição substancial de aplicativos de Multimídia e Computação em Nuvem, em especial de processamento baseado em Data Centers. Data Centers podem ser definidos como instalações físicas onde vários servidores e equipamentos de comunicação estão presentes devido às suas necessidades ambientais e de segurança comuns, bem como a facilidade de manutenção (Barroso et al., 2013).

De acordo com estudos analíticos apresentados no relatório Visual Networking Index 2017-2022 da Cisco (Cisco, 2018), em 2019 o tráfego global da Internet excederá 200 Exabytes por mês, atingindo 396 Exabytes por mês em 2022. Além disso, se considerarmos especificamente o cenário de tráfego de vídeo do consumidor transmitido na Internet, será observado que em 2019 o tráfego de vídeo será responsável por aproximadamente 82% do total e chegará a 84% em 2022.

Para atender às demandas crescentes de tráfego desses serviços, como o vídeo *Ultra HD*, há uma necessidade urgente de usar com eficiência os recursos disponíveis em uma infraestrutura de rede, como por exemplo a largura de banda e *slot* de frequência de uma Rede Óptica Elástica (Elastic Optical Network - EON) (Jinno et al., 2009; Sone et al., 2011; Shen et al., 2016) e os recursos de servidores réplicas de um Data Center (Barroso et al., 2013). Estes requisitos irão forçar os provedores de serviços, especialmente os que lidam com streaming de vídeo, a considerar novos mecanismos para o tratamento do tráfego de rede.

O tratamento inadequado do tráfego é um dos fatores que afetam o desempenho de esquemas como Sobrevivência de Rede (Network Survivability), Engenharia de Tráfego (Traffic Engineering - TE), Qualidade de Serviço (Quality of Service - QoS), Controle de Acesso Dinâmico e outros, produzindo problemas de dimensionamento de infraestrutura (Tongaonkar et al., 2015). Assim, a Classificação de Tráfego é um

mecanismo chave para o tratamento do tráfego, fornecendo uma base de conhecimento para determinar os níveis de desempenho exigidos pelos aplicativos.

Os métodos mais comuns para classificação de tráfego são a Inspeção Profunda de Pacotes (Deep Packet Inspection - DPI) e a Classificação Baseada em Porta. Esses métodos estão se tornando obsoletos à medida que cada vez mais tráfego é criptografado e os aplicativos estão usando portas dinâmicas de outros aplicativos populares (Shi et al., 2017). Um método alternativo para classificação de tráfego é o Aprendizado de Máquina (Machine Learning - ML), que usa os recursos estatísticos dos fluxos de tráfego de rede para resolver os problemas fundamentais de DPI e Classificação Baseada em Porta para fluxos criptografados.

A classificação de tráfego de rede é uma abordagem essencial para o gerenciamento de infraestrutura e para garantir a QoS das diferentes aplicações. Na realidade, um procedimento preciso de classificação de tráfego permite o gerenciamento eficiente dos recursos de rede existentes, permitindo, assim, esquemas mais precisos e robustos de alocação dos mesmos (Liu et al., 2018; Sun et al., 2018c).

Conceitos como redes com reconhecimento de aplicações ou programação de aplicações com reconhecimento de rede, também surgem como propostas para superar as limitações da classificação de tráfego. Nessa direção, com uma arquitetura de controle centralizada e promissora, o paradigma das Redes Definidas por Software (Software Defined Networks - SDN) (FOUNDATION, 2012), incluindo o protocolo OpenFlow (McKeown et al., 2008), está surgindo como um caminho para a realização de uma interação entre redes e aplicações. A arquitetura centralizada SDN faz com que o controlador tenha uma visão central da rede (Huang et al., 2017). A característica de programação de rede permite que os aplicativos programem a rede. Nesse sentido, o objetivo da SDN é simplificar o gerenciamento da rede, promover a inovação, melhorar a utilização de recursos e otimizar o desempenho.

Nos últimos anos, a SDN trouxe novas oportunidades para classificação de tráfego e seleção de características (Yan e Yuan, 2018). A visão global dos controladores em SDN é simples para extrair características estatísticas do tráfego de rede dos switches. Devido ao *design* do mecanismo do protocolo OpenFlow, ele pode ser personalizado para reunir recursos de fluxo e indicar políticas de encaminhamento para cada switch.

Técnicas de classificação de tráfego em SDN foram investigadas, propostas e desenvolvidas. Em (Qazi et al., 2013), os autores propõem um *framework* denominado de **Atlas** para classificação de tráfego em rede sem fio aplicando a SDN em ambiente com sistema operacional *Android*. No framework, é adotado o algoritmo de Árvore de Decisão C5.0 para a classificação. Uma abordagem de *crowd sourcing* é usada para coletar dados verdadeiros dos dispositivos finais. Os dados coletados são usados para

treinar a árvore de decisão. O modelo treinado é capaz de identificar as aplicações móveis dos fluxos de tráfego. Os resultados da simulação demonstram que a precisão média de classificação do **Atlas** é superior a 94% para os 40 principais aplicativos no Google Play.

O trabalho proposto por (Da Silva et al., 2015) apresenta um *framework* para identificar, estender e selecionar conjuntos de características de fluxo para classificação de tráfego de redes baseadas no protocolo OpenFlow. A coleta é obtida através de contadores nativos do OpenFlow, bem como de estatísticas matemáticas e Transformada Discreta de Fourier (DFT), sendo que o algoritmo de ML usado no *framework* proposto é o SVM. Para encontrar um subconjunto ideal de características para classificação de tráfego, eles adotam a Análise de Componentes Principais (PCA) e o Algoritmo Genético (GA). A principal contribuição deste trabalho é que ele fornece um conjunto abundante de características de fluxo de tráfego na SDN e aprimora os recursos para encontrar as características mais relevantes pelos algoritmos PCA e GA. No entanto, as características são de nível de fluxo e exigem computação complicada devido à DFT. Portanto, leva ao fracasso em lidar com a classificação de tráfego em tempo real. (Xiao et al., 2016) introduz um método de classificação de tráfego na SDN, que é baseado em uma técnica de ML não supervisionada denominada *clustering* espectral. A solução extrai características de fluxo examinando as tabelas de fluxo no controlador SDN e as características são agrupadas com análise espectral. Os experimentos foram conduzidos em um conjunto de dados de um Data Center, e os protocolos utilizados foram HTTP, SMTP, SSH, P2P e DNS. O método proposto foi comparado com o método tradicional KMEANS e os resultados demonstram que o método proposto supera o tradicional em termos de precisão e *recall*. No entanto, o trabalho não descreve as características de maneira detalhada usadas para classificação. No trabalho (Amaral et al., 2016) é proposto uma arquitetura para coletar dados de tráfego em uma rede corporativa usando o protocolo OpenFlow, que pode ser aplicada em redes SDN, bem como em redes legadas. Para reduzir a redundância de características, os autores aplicam o algoritmo de Análise de Componentes Principais (PCA) para encontrar os componentes ideais. Os autores implementam três métodos de ML supervisionado: Random Forests (RF), Stochastic Gradient Boosting (SGB) e Extreme Gradiente Boosting (EGB). Este método reúne informações dos primeiros cinco pacotes, o que gerou uma razoável acurácia média de 90% para cada aplicação identificada. Em (Wang et al., 2016) é apresentado um esquema de classificação de tráfego com reconhecimento de QoS em SDN. A abordagem adotada classifica o tráfego em classes, de acordo com os diferentes requisitos de QoS das aplicações. O esquema combina a técnica de DPI para a rotulagem dos fluxos com um algoritmo semi-supervisionado *Laplacian SVM* para

classificação das classes. Os fluxos foram categorizados em quatro classes de QoS: (1) Voice/Video Conference; (2) Interactive Data; (3) Streaming e (4) Bulk Data Transfer. Os resultados da simulação mostram que a abordagem de classificação proposta pode fornecer um desempenho promissor. A precisão de classificação excede 90%. Contudo, os autores não apresentam os resultados para o custo computacional da classificação, além de fazer a classificação do tráfego com granulação grossa para QoS, em vez de fazer classificação por granuação fina. Os autores em (Parsaei et al., 2017) propõem uma estrutura (*testbed*) para classificação de tráfego *on-line* utilizando o paradigma SDN. A classificação é baseada em características do fluxo do tráfego de acordo com cinco protocolos da camada de aplicação: FTP, HTTP, mensagens instantâneas, streaming de vídeo e peer-to-peer. Foram implementadas quatro variantes de Rede Neural para classificar o tráfego de acordo com o tipo de protocolo: feedforward, Multilayer Perceptron (MLP), NARX (Levenberg-Marquardt) e NARX (Naive Bayes). Estes quatro cenários de classificação, respectivamente, obtêm uma acurácia de 95,6%, 97%, 97% e 97,6%. Contudo, a estrutura proposta tem que coletar e calcular informações com base no fluxo total do tráfego. Tais requisitos podem impedir a aplicação de tráfego antecipado na classificação.

Um aspecto interessante, quando se aplica técnicas e algoritmos de ML na classificação de tráfego, é que seus resultados são divididos em duas categorias, de acordo com o nível de classificação: granulação grossa e granulação fina. A classificação de granulação grossa identifica o fluxo de tráfego em várias classes com base no tipo de tráfego aproximado (navegador da web, transferência em massa, VoIP e assim por diante). No entanto, é cada vez mais importante classificar o tráfego de rede com resultados refinados. A classificação fina visa distinguir a aplicação individual ou o modelo de função, em vez de classificá-los em uma classe de tráfego aproximada, que é favorável à análise da composição do tráfego e dos perfis de usuário. Nesse sentido, os trabalhos (Da Silva et al., 2015; Xiao et al., 2016; Wang et al., 2016; Parsaei et al., 2017) classificam os tráfegos com granulação grossa. Já os trabalhos (Amaral et al., 2016; Qazi et al., 2013), aplicam técnicas de ML com classificação de granulação fina. Contudo, existem poucos trabalhos que abordam a classificação em redes de computadores de maneira fina (Boutaba et al., 2018; Xie et al., 2018).

Outro aspecto é que a maioria das pesquisas citadas classificam o tráfego em um conjunto de estatísticas de fluxos armazenados de maneira *offline*. A alta complexidade de tempo e a sobrecarga de processamento são dois desafios da classificação de tráfego *on-line*. As abordagens atuais também causam sobrecarga considerável no sistema. Os resultados apresentados nos trabalhos (Da Silva et al., 2015; Xiao et al., 2016; Amaral et al., 2016; Parsaei et al., 2017) demonstraram que os métodos adotados para classifica-

ção têm que coletar informações de todo o fluxo, fazendo com que esse requisito possa impedir suas aplicações em um cenário real de classificação de tráfego, pois geram um alto custo computacional para classificação *on-line*.

No estudo (Cisco, 2018), sobre o tráfego global na nuvem, é previsto que o tráfego global dos Data Centers se multiplicará cerca de três vezes entre 2017-2022, passando de 3,4 Zettabytes para 10,04 Zettabytes, sendo este tráfego o de maior magnitude entre os estudados. No que diz respeito a escalabidade, os Data Centers tradicionais não a conseguem para o seu armazenamento e nem para carga computacional, a não ser através da compra e instalação de mais equipamentos e recursos. Data Centers em nuvem são escaláveis na medida em que os seus sistemas são demandados, podendo ter capacidade ilimitada de crescimento, de acordo com os projetos de cada provedor.

Nesta mesma direção, surge o problema de atender não só o volume futuro de tráfego nos Data Centers, mas atender também o tráfego atual e suas oscilações com essa nova demanda por taxas de transmissão de alta velocidade requisitada pelos serviços na Internet (Cisco, 2018). Um dos principais problemas gerados pelo aumento do volume é em relação ao uso eficiente da taxa de transferência, vazão, para atender as requisições. Duas formas de resolver este problema são comumente empregadas. Uma primeira abordagem é aumentar a capacidade dos enlaces físicos, contudo essa abordagem tem um custo alto de implantação e não é escalável, pois os enlaces físicos podem chegar ao limite de velocidade rapidamente e assim será necessário a implantação de outra tecnologia (Barroso et al., 2013). Uma segunda abordagem é a implementação e configuração de servidores réplicas para aumentar a capacidade para atender mais serviços simultaneamente. Porém, definir servidores réplicas sem o devido tratamento do tráfego, poderá acarretar na subutilização ou superutilização dos recursos físicos disponibilizados por tais máquinas. Os servidores réplicas, na sua grande maioria, são configurados para atender requisição de serviços das aplicações de acordo com as portas associadas no cabeçalho TCP/UDP e vinculadas aos números de portas conhecidas e direcionadas por um servidor dedicado de balanceamento de carga. Atualmente, algumas aplicações robustas, como streaming de vídeo, são transmitidas principalmente usando os protocolos HTTP e HTTPS, cujas as portas, 80 e 443, são usadas por praticamente todos os aplicativos Web, fazendo com que a precisão para identificar os fluxos da Internet seja baixa. (Madhukar e Williamson, 2006).

Cada uma das abordagens apresentadas como propostas para solução do problema citado acima traz beneficios importantes, como também um alto risco em sua escalabilidade e operacionalização dos recursos, podendo não atender a todos os perfis de instituições. Porém, o uso combinado das duas abordagens, aumento da capacidade da vazão dos enlaces e a utilização de vários servidores réplicas, levam na direção

de uma solução mais adequada. Isso inclui planejamento da capacidade da taxa de transferência e provisionamento de carga, como exemplos.

Para serem sustentáveis em um ambiente competitivo, as operadoras de redes devem adotar um gerenciamento da rede de maneira eficiente e acessível (Xie et al., 2018). A classificação de tráfego é fundamental para que as operadoras executem uma ampla variedade de atividades de operação e gerenciamento de rede. A classificação de tráfego, utilizando aprendizado de máquina, exige a capacidade de associar com precisão o tráfego de rede a classes de interesse pré-definidas. Além disso, os principais avanços tecnológicos, com capacidade de programação em rede por meio de SDN, promovem a aplicabilidade do aprendizado de máquina em redes (Boutaba et al., 2018).

Embora o ML tenha sido amplamente aplicado a problemas de reconhecimento de padrões, síntese de fala e detecção de *outliers*, sua implantação para gerenciamento e operações em redes de maneira bem-sucedida ainda é bem limitada (Boutaba et al., 2018). Os principais obstáculos incluem quais dados podem ser coletados e quais ações podem ser realizadas em dispositivos de rede legadas.

A capacidade de programar a rede SDN mitiga esses obstáculos (Kreutz et al., 2015; Xie et al., 2018). Avanços recentes no paradigma SDN permitem técnicas flexíveis e adaptativas para classificação de tráfego. A SDN oferece mecanismos integrados para coleta de dados por meio do protocolo OpenFlow. A compreensão sobre ML pode ser usada para auxiliar na automação de tarefas de operação e gerenciamento de rede. Portanto, é interessante e não trivial aplicar técnicas de ML para problemas tão diversos e complexos em redes. Isso torna a aplicação de ML em rede uma área de pesquisa interessante e que requer um adequado entendimento das técnicas de ML e dos problemas de rede (Boutaba et al., 2018).

A motivação para este trabalho de tese foi a de propor uma solução inovadora para os problemas causados em Data Centers pelo crescente aumento do tráfego na Internet (Cisco, 2018), através do balanceamento de carga de entrada em um Data Center com arquitetura de duas camadas (*Two Tier*) (Al-Fares et al., 2008) (Barroso et al., 2013). O tráfego de entrada das requisições de serviço dos clientes é dividido por enlaces distintos para servidores réplicas, obedecendo a uma política de balanceamento de carga baseado em classificação de tráfego. A proposta evitará perdas drásticas da QoS para os clientes pela falta de atendimento ou excessiva demora na transferência dos serviços oferecidos pela rede, além de otimizar os recursos de vazão e diminuição da perda de pacotes (*packet-loss*).

O trabalho é centrado em duas motivações. A primeira é propor um módulo inovador de classificação de tráfego de vídeo baseado em ML, como uma solução para

rotulagem de pacotes com granularidade fina. A classificação dos tráfegos de maneira fina visa distinguir as aplicações individualmente, favorecendo a análise da composição do tráfego e dos perfis de usuário, além de incorrer em um custo computacional muito menor do que a abordagem baseada em DPI, por exemplo. Com isso, poderá ser utilizada a eficiência da Engenharia de Tráfego. O serviço de encaminhar um grande volume de tráfego dividido por vários enlaces é um método simples e eficiente de Engenharia de Tráfego, dentre vários disponíveis (Leduc et al., 2006). Um desses métodos de Engenharia de Tráfego é chamado de balanceamento de carga. Sendo assim, o módulo de classificação de tráfego baseado em ML atuará como método de balanceamento de carga, fazendo o tratamento dos tráfegos e fornecendo uma base de conhecimento para determinar os níveis de desempenho exigidos pelos aplicativos. Com a ajuda do módulo de classificação o balanceamento de carga poderá lidar com diferentes serviços e alocar recursos de rede de maneira mais eficiente.

Outra motivação é a nova abordagem fornecida pelo paradigma SDN com várias vantagens para o gerenciamento da rede, através do protocolo OpenFlow (Kreutz et al., 2015; Farhady et al., 2015; Xie et al., 2018). Desta forma podem ser oferecidas melhores abstrações e plataformas para a inovação, permitindo propor novos serviços que antes não eram possíveis com switches comerciais e hardware não proprietário e de alto custo (McKeown et al., 2010). Uma nova abordagem para o balanceamento de carga pode ser implementada com SDN.

A técnica tradicional de balanceamento de carga tem um hardware dedicado, o balanceador de carga, para gerenciar o tráfego dos clientes com diferentes servidores. A Figura 1.1 mostra um modelo simples de sistema típico de balanceamento de carga. Conforme mostrado na figura, os clientes estão conectados ao servidor de balanceamento, que já está conectado ao switch gateway. Quando o balanceador de carga recebe uma solicitação de qualquer cliente, ela a encaminha para um servidor capaz de atendê-la. Portanto, o balanceador de carga deve acompanhar todas as sessões, sendo ainda capaz de realizar as funções de Network Address Translation (NAT) e Domain Name System (DNS) (Senthil Ganesh e Ranjani, 2015).

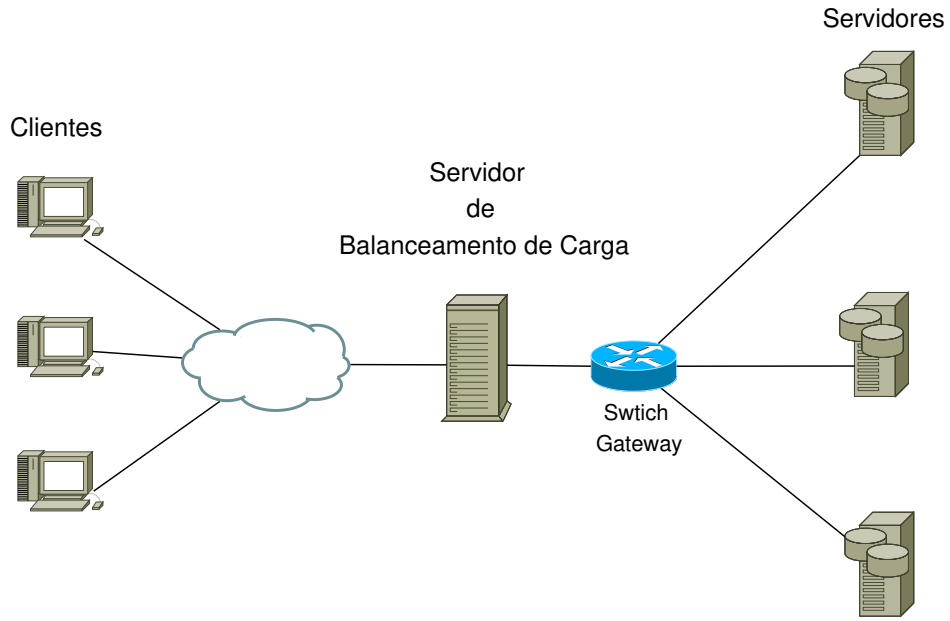


Figura 1.1: Modelo de balanceamento de carga tradicional.

A desvantagem de ter um hardware dedicado para fins de balanceamento de carga é que ele é considerado uma solução cara e não flexível. Além disso, sofre do problema de ter o único ponto da falha e de representar um gargalo para o sistema inteiro (Senthil Ganesh e Ranjani, 2015; Qilin e Weikang, 2015).

Já o balanceamento de carga com base em SDN não necessita de um hardware dedicado à rede. A Figura 1.2 mostra um modelo simples de sistema de balanceamento de carga baseado em SDN.

Na SDN, o controlador tem uma visão de rede global, o que facilita a coleta e a análise de tráfego. Assim, as abordagens baseadas em ML são geralmente implementadas no controlador. O controlador da rede baseada em SDN é conectado a todos os switches e, ao adicionar ao controlador o módulo de classificação proposto neste presente trabalho, será possível gerenciar solicitações de qualquer cliente e direcioná-las a um enlace (ou servidor) específico, de acordo com as informações classificadas pelo controlador.

O módulo de classificação de tráfego de streaming de vídeo é usado no balanceador de carga, sendo este capaz de adicionar, excluir ou modificar as entradas da tabela de fluxo localizadas nos switches baseados no OpenFlow. Assim, comparado ao balanceamento de carga tradicional, o balanceamento de carga baseado em SDN tem uma implementação mais simples, com melhor desempenho (Senthil Ganesh e Ranjani, 2015; Qilin e Weikang, 2015; Xie et al., 2018).

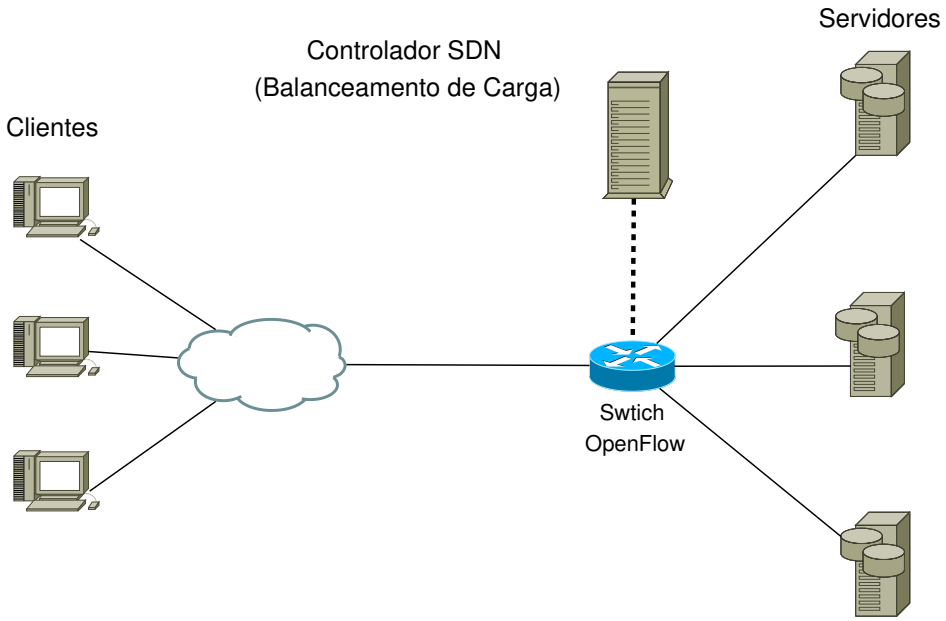


Figura 1.2: Modelo de balanceamento de carga baseado em SDN.

1.2 Objetivos

1.2.1 Geral

O principal objetivo deste trabalho foi o de desenvolver um classificador de tráfego para ambiente de Redes Definidas por Software (SDN), com uma abordagem inovadora de classificação de tráfego de rede em tempo real, para utilização em esquemas de Engenharia de Tráfego em Data Centers. Essa abordagem inovadora buscou proporcionar uma maior acurácia (precisão), ter uma boa taxa de acertos por classe e respeitar os parâmetros de complexidade computacional para o tráfego de rede multiclasse. Ou seja, obter o melhor *trade-off* entre o desempenho geral e os custos computacionais da classificação, para ser aplicado como balanceador de carga de entrada em um Data Center com arquitetura de duas camadas.

1.2.2 Específicos

A. Desenvolver um classificador multiclasse baseado em aprendizado de máquina com granulação fina.

B. Propor um novo algoritmo de aprendizado de máquina baseado no modelo Naive Bayes, com uma nova abordagem quanto ao relaxamento da hipótese de independência entre os atributos.

C. Propor e implementar scripts Python personalizados para captura, geração de dataset e extração de características.

D. Propor e implementar um classificador de tráfego em tempo real baseado no paradigma SDN.

E. Propor e implementar um balanceador de carga utilizando o classificador de tráfego baseado no paradigma SDN.

F. Propor uma análise comparativa com ênfase no aspecto da complexidade computacional com outros algoritmos de aprendizado de máquina.

1.3 Estrutura do Texto

Este texto está organizado em seis capítulos da seguinte forma. O presente capítulo apresentou a motivação e a justificativa deste trabalho. O Capítulo 2 apresenta uma revisão concisa da caracterização e classificação do tráfego. No Capítulo 3 é apresentado o conceito e a arquitetura do paradigma do plano de controle baseado nas Redes Definidas por Software. O Capítulo 4 apresenta a abordagem metodológica adotada para o módulo de classificação de tráfego de streaming de vídeo baseado em aprendizado de máquina. O módulo adota uma nova abordagem para o relaxamento da hipótese de independência entre os atributos do algoritmo Naive Bayes. O Capítulo 5 descreve a integração do módulo de classificação de tráfego de streaming de vídeo a uma arquitetura SDN em um cenário de rede em Data Center. O Capitulo 6 traz os principais pontos abordados e contribuições desse trabalho. Também, são apresentadas algumas sugestões de trabalhos futuros que não foram investigados, mas que merecem ser consideradas.

Capítulo 2

Caracterização e Classificação de Tráfego

2.1 Introdução

Neste capítulo, é apresentada uma revisão compacta relacionada ao escopo deste trabalho em dois contextos: caracterização do tráfego de vídeo e uma visão histórica do problema de classificação de tráfego. Esta seção enfoca a classificação *on-line*.

2.2 Caracterização de Tráfego de Vídeo

Os aplicativos de streaming de vídeo estão se tornando uma proporção cada vez maior do tráfego de Internet, com previsões de que ele atingirá mais de 84% do tráfego global até 2022. No entanto, poucos estudos tentam entender as características da rede desse tráfego.

Um desses trabalhos (Rao et al., 2011a) investiga as características da rede dos dois serviços de streaming mais populares: *Netflix* e *YouTube*. É mostrado que a estratégia adotada para streaming varia com o aplicativo (um navegador da Web ou aplicativo nativo) e protocolos (*Silverlight*, *Flash* ou *HTML5*) usados. A Figura 2.1 ilustra os comportamentos desses serviços durante uma típica sessão de streaming de vídeo.

Conforme ilustrado na Figura 2.1, um streaming de vídeo começa com uma fase de carregamento (*buffering*) e é seguido por uma fase de estado estacionário (*steady state*). Além disso, os ciclos ON-OFF são usados durante o estado estacionário para limitar a taxa de *download*.

É importante observar que, durante a fase de armazenamento em buffer, a transferência de dados é limitada pela largura de banda disponível e que a inclinação da reta nesse estágio é equivalente a esse valor. O vídeo começa a tocar quando uma quantidade de dados suficiente está disponível em seu *buffer*.

Outro aspecto é que, na fase estacionária, a taxa média de *download* é ligeiramente superior à taxa de codificação de vídeo. A razão entre essas duas taxas é conhecida como taxa de acumulação (*acummulation rate*) e os resultados apresentados neste trabalho indicam que seu valor deve ser maior que um, para permitir a reprodução do vídeo sem interrupção.

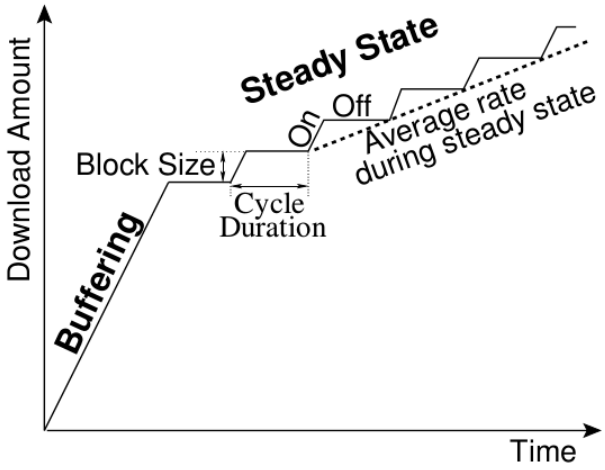


Figura 2.1: Comportamento genérico durante uma sessão de streaming de vídeo típica (Rao et al., 2011a).

De acordo com (Rao et al., 2011a), os provedores citados adotam três estratégias diferentes:

1. **Sem ciclos ON-OFF:** Todos os dados são transferidos durante a fase de carregamento (buffering), não sendo observada uma fase estacionária. Pode ser considerada uma simples transferência de arquivo.

2. **Ciclos ON-OFF curtos:** Os blocos transferidos periodicamente nos ciclos possuem tamanho menor que 2,5 MB e são seguidos por períodos inativos (*idle*). O objetivo dessa estratégia é manter a taxa de acumulação um pouco superior a um.

3. **Ciclos ON-OFF longos:** Os blocos transferidos periodicamente remontam a execução periódica da fase de carregamento seguida por um grande período inativo. A principal diferença entre essa estratégia e a anterior é a quantidade de dados transferida em um ciclo, que é superior a 2,5 MB.

Para fazer esse estudo, (Rao et al., 2011a) utilizou seis bases de dados. Para o *YouTube*, foram quatro bases: *YouFlash*, *YouHtml*, *YouHD*, e *YouMob*. As três primeiras contemplam vídeos transferidos por *Web browsers*, sendo as duas primeiras para vídeos SD com seus respectivos protocolos em uso na época: *Flash* e HTML5. A terceira base

é referente a vídeos de alta resolução transferidos via *Flash* e a quarta contém dados de captura de transferências de vídeo por dispositivos móveis, utilizando o aplicativo de propriedade do *YouTube*. Para o *Netflix*, foram duas bases: *NetPC* e *NetMob*. A primeira contempla vídeos transferidos pelo *browser* e a segunda por aplicativo móvel. Essas bases de dados assim como *scripts* úteis para seu processamento estão disponíveis para *download* em (Rao et al., 2011b).

2.3 Classificação de Tráfego

Com relação à classificação de tráfego, as primeiras técnicas examinaram as portas de comunicação no cabeçalho TCP / UDP e as associaram aos números de porta conhecidos (IANA, 2019) para identificar quais aplicativos originaram o tráfego (Blake et al., 1998). No entanto, o tráfego de vídeo hoje é transmitido principalmente usando os protocolos HTTP e HTTPS, cujas portas, 80 e 443, são usadas por praticamente todos os aplicativos da Web. Por esse motivo, essa técnica seria muito pouco eficiente. Nos últimos anos, vários resultados de experimentos de avaliação mostraram que os métodos baseados em portas não são muito eficientes. (Moore e Papagiannaki, 2005) observaram que não há mais de 70% de precisão para técnicas baseadas em portas usando a lista (IANA, 2019). (Madhukar e Williamson, 2006) descobriram que as técnicas baseadas em porta não conseguem identificar de 30 a 70% dos fluxos de tráfego da Internet.

Para superar a limitação das técnicas de classificação baseadas em portas, foram propostas técnicas baseadas em carga útil. As abordagens de carga útil também são conhecidas como DPI (*Deep Packet Inspection*). Elas classificam o tráfego inspecionando cargas úteis de pacotes e comparando com as assinaturas conhecidas de protocolos (Sen et al., 2004; Finsterbusch et al., 2014; Goo et al., 2016; Fu et al., 2017). Várias ferramentas de DPI têm sido amplamente utilizadas, como o filtro L7 (L7-Filter, 2009) e o OpenDPI (OpenDPI, 2012). Técnicas baseadas em DPI podem executar a classificação de tráfego com precisão, mas levam a uma complexidade significativa e alto custo de computação. E as técnicas de DPI podem ser difíceis ao lidar com protocolos criptografados ou proprietários. (Gringoli et al., 2009) avaliaram o desempenho dos métodos DPI, e os resultados experimentais mostram que as ferramentas DPI, como o filtro L7, apenas classificam corretamente 67,73% dos bytes no conjunto de dados UNIBS (UniBS, 2011) e 58,79% dos bytes no conjunto de dados POLITO (di Torino, 2019), respectivamente. Além disso, os métodos DPI exigem assinatura de manutenção manual para manter-se atualizado com a semântica de aplicativos, enquanto às vezes é difícil ou impossível obter assinaturas com a evolução dos aplicativos de rede.

Portanto, as técnicas de aprendizado de máquina surgiram como uma alternativa para a classificação de tráfego, permitindo a busca por padrões estruturais dentro da rede.

2.3.1 Classificação de Tráfego baseado em Aprendizado de Máquina

Com base nas propriedades estatísticas extraídas dos fluxos e subfluxos de rede, (Li et al., 2013; Nguyen e Armitage, 2006, 2008; Wang e Yu, 2009; Zhang et al., 2009) realizam análises comparativas usando algoritmos ML para identificar as variáveis mais representativas e úteis para a tarefa de classificação. No entanto, esses estudos apresentam limitações relacionadas à caracterização do tamanho ideal de um subfluxo para as classes de aplicação utilizadas e à quantidade de classes de aplicação a serem classificadas, uma vez que a maioria delas utiliza classificação com granulação grossa. Além disso, esses estudos não exploram técnicas de minimização para o número de variáveis utilizadas.

Trabalhos recentes abordam a classificação de tráfego em ambientes criptografados. Os autores em (Alshammari e Zincir-Heywood, 2009) avaliam cinco algoritmos ML (AdaBoost, DVM, Naive Bayes, RIPPER e C4.5) para classificar dois tipos de tráfego criptografado, *SSH* e *Skype*. Os algoritmos C4.5 e RIPPER apresentam o melhor desempenho geral entre os diferentes conjuntos de dados e classes estudados. Em outro trabalho (Alshammari e Zincir-Heywood, 2015), os autores apresentam uma abordagem ML para identificar o VoIP no tráfego de rede criptografado. Esta abordagem compara três algoritmos ML supervisionados diferentes (C5.0, AdaBoost e Genetic Programming - GP). Os resultados mostram que o C5.0 obteve o melhor desempenho entre os três algoritmos estudados. Em (Datta et al., 2015), os autores propõem uma técnica baseada no comportamento do aplicativo usando extração de características e classificação de tráfego em um ambiente criptografado. Para os experimentos, o tráfego de rede é coletado e dividido em quatro classes diferentes, Google Hangout, Google Plus, Gmail, e não Hangout. Três algoritmos ML são comparados (Naive Bayes, AdaBoost e a Árvore de Decisão J48) usando três cenários de teste: somente Hangout e não-Hangout, incluindo tráfego do Gmail, Gmail e Google Plus. No primeiro cenário, J48 e AdaBoost obtêm os melhores resultados. No segundo, AdaBoost e Naive Bayes são os melhores. No último cenário, AdaBoost e J48 alcançam os melhores resultados.

Do ponto de vista da classificação de tráfego on-line, existem trabalhos que aplicam o aprendizado de máquina para classificar o tráfego em tempo real, incluindo o streaming de vídeo (Li et al., 2013; Nguyen e Armitage, 2006, 2008; Wang e Yu, 2009; Zhang et al., 2009). No entanto, a grande maioria dos conjuntos de dados está desatualizada devido

à rápida evolução dos mecanismos utilizados, como é o caso das bases usadas em (Rao et al., 2011a). No momento da publicação, em 2011, o *YouTube* usava o *Flash* principalmente para exibir seus vídeos, enquanto o *Netflix* preferia o *Silverlight*. Nos últimos anos, ambos adotaram o HTML5, principalmente porque é gratuito e aberto. Portanto, embora os conjuntos de dados usados em (Rao et al., 2011a) ainda estejam disponíveis (Rao et al., 2011b), há uma grande probabilidade de que alguns deles estejam desatualizados devido à rápida evolução dos protocolos HTML5.

Os autores em (Gomes e Madeira, 2012) propõem um agente de classificação de tráfego em tempo real para redes virtualizadas baseadas em classes de QoS e usando classificação baseada em ML. O trabalho mostra que, dentre as técnicas de classificação avaliadas, o agente Naive Bayes possui uma acurácia de 95% e o menor tempo de classificação. O trabalho apresentado em (Shi e Biswas, 2016) cita uma característica interessante chamada Packet Arrival Interval (PAI), que pode ser muito útil para um cenário no qual os fluxos já estão separados. O PAI também é interessante porque, de acordo com o modelo apresentado na Figura 2.1, ele pode detectar diferentes tempos ON-OFF para cada provedor, o que já foi abordado em (Rao et al., 2011a). Um trabalho mais recente (Sun et al., 2018b) estuda a classificação de tráfego de rede com base no Aprendizado por Transferência (*Transfer Learning*). No estudo, o TrAdaBoost é aplicado para abordar a classificação de tráfego de rede de várias classes como uma estrutura de aprendizado. O modelo de máxima entropia (Maxent) é adotado como o classificador base no modelo de aprendizado de transferência. Os resultados experimentais mostram que um bom desempenho de classificação é obtido com base no modelo de aprendizagem por transferência. A fim de abordar duas limitações principais na tarefa de classificação de tráfego do Modelo SVMs (*Support Vector Machines*), os autores de (Sun et al., 2018a) usam o modelo SVMs incrementais (ISVM), apresentado em (Syed et al., 1999) para reduzir o alto custo de treinamento de memória e CPU, e identificar classificadores de tráfego com alta frequência e atualizações rápidas. Além disso, uma versão modificada do modelo ISVM com fator de atenuação, chamado AISVM, é proposta para utilizar informações valiosas nos conjuntos de dados de treinamento anteriores. Os resultados experimentais comprovaram a eficácia dos modelos ISVM e AISVM na classificação de tráfego.

Em (Yu et al., 2018), os autores propõem uma estratégia de alocação de recursos baseada em aprendizagem profunda em redes ópticas intra-Data Center para melhorar a precisão da previsão de tráfego. Com base na estratégia, é proposto um algoritmo que pode fornecer alocação de recursos altamente confiável para Data Centers, com previsão de tráfego de alta precisão e melhorar a utilização de recursos de tráfego de alta largura de banda e múltiplas variedades. Em outro trabalho (Yao et al., 2018),

os autores apresentam um novo modelo de cálculo de diafonia espacial baseado no aprendizado de máquina com o método de propagação de feixes (BPM), denominado CEM-BPM-ML, para o esquema de alocação de recursos em Redes Ópticas Elásticas com Multiplexação por Divisão Espacial. Em vista disso, três algoritmos ML diferentes são usados para a predição de diafonia espacial, que são Levenberg-Marquardt (LM), Bayesian Regularization (BR) e o Scaled Conjugate Gradient (SCG). A simulação indica que o algoritmo LM apresenta melhor desempenho na medição dos valores de regressão de treinamento e consumo de tempo. Em (Lopez-Martin et al., 2018), os autores apresentam um classificador para prever a Qualidade da Experiência (*Quality of Experience* - QoE) de vídeo com base nas informações extraídas diretamente dos pacotes de rede usando um modelo de aprendizagem profunda. O classificador proposto é baseado em uma combinação de uma Rede Neural Convolucional (CNN), Rede Neural Recorrente e Classificador de Processo Gaussiano. Com base em uma comparação detalhada, o modelo proposto oferece melhores métricas de desempenho do que algoritmos alternativos de aprendizado de máquina e pode ser usado como uma função de monitoramento de QoE na computação de borda.

Existem trabalhos que abordam uma questão difícil na classificação de tráfego, que é como obter um modelo de classificador de tráfego baseado em características estatísticas com alta precisão e usando um pequeno conjunto de dados de treinamento. Os autores de (Zhang et al., 2013a) propõem um novo esquema de classificação de tráfego de rede, onde os fluxos de tráfego são descritos usando as características estatísticas discretizadas e as informações de correlação de fluxo são modeladas por Bag-of-Flow (BoF). Além disso, um novo modelo de classificação de tráfego baseado em BoF é proposto para agregar as previsões Naive Bayes (NB) dos fluxos correlacionados. Os resultados mostram uma solução promissora, capaz de alcançar classificação de tráfego de alto desempenho sem a demorada rotulagem de amostras de treinamento. Em outro trabalho (Zhang et al., 2013b), os autores apresentam uma nova abordagem não paramétrica para a classificação de tráfego usando informações de correlação, a fim de melhorar o desempenho do classificador quando o tamanho dos dados de treinamento for pequeno. Para a abordagem, são implementados quatro métodos de classificação que incorporam o Bag-of-Flows (BoF) para modelar as informações de correlação nos fluxos de tráfego. O método de classificação baseado no vizinho mais próximo (NN), que representa os métodos de classificação convencionais, foi escolhido para comparação com a abordagem proposta. Os resultados demonstraram que, a abordagem proposta pode ser usada em uma ampla gama de aplicações, como reconhecimento automático de aplicações desconhecidas a partir do tráfego de rede capturado e mineração de dados semi-supervisionada para processamento de pacotes de rede. Em cenários

com aplicações *zero-day* anteriormente desconhecidas em sistemas de classificação de tráfego, os autores em (Zhang et al., 2015) propõem um esquema de classificação de tráfego estatístico robusto (RTC). O RTC pode descobrir aplicações desconhecidas em sistemas de classificação de tráfego, combinando técnicas de aprendizado de máquina supervisionadas e não supervisionadas e usando correlação de fluxo. Um novo estudo quantitativo, baseado na teoria da probabilidade, é aplicado para mostrar como a correlação pode beneficiar a classificação de tráfego melhorando o desempenho da classificação. O esquema RTC é comparado com quatro métodos de aprendizado de máquina: floresta aleatória, classificação baseada em correlação, *clustering* semi-supervisionado e SVM de uma classe. Os resultados demonstram que o RTC superou significativamente os outros quatro métodos, melhorando efetivamente as precisões de classes conhecidas quando os aplicativos *zero-day* estão presentes. Outro aspecto nos trabalhos (Zhang et al., 2013a) (Zhang et al., 2013b) (Zhang et al., 2015) é que eles usam traços do tráfego de rede do mundo real em seus conjuntos de dados, coletados entre 2006 e 2010.

A classificação do tráfego de rede também está sendo considerada, nos últimos anos, uma ferramenta valiosa na segurança cibernética. Do ponto de vista da detecção e prevenção contra ameaças internas, no *survey* (Liu et al., 2018), os autores revisam esquemas e sistemas propostos que abordam ameaças internas a partir das perspectivas dos três tipos mais comuns de *insider*: traidor (*traitor*), mascarador (*masquerader*) e criminoso não intencional (*unintentional perpetrator*). Os esquemas abordados concentram-se sua análise com base em dados de *hosts*, rede e contexto. Para cada trabalho citado, sua capacidade contra ameaças internas, como ele extrai características das fontes de dados e como um algoritmo de aprendizado de máquina é aplicado para tomar decisões são revisados e também comparados e contrastados. Em outro survey (Sun et al., 2018c), os autores revisam artigos do ponto de vista da previsão de incidentes de cibersegurança orientado por dados, que aplicam principalmente técnicas de aprendizado de máquina, mineração de dados, aprendizado profundo e mineração de grafos. Os artigos coletados são classificados em seis categorias, de acordo com conjuntos de dados utilizados: relatórios da organização e conjunto de dados, conjunto de dados de rede, conjunto de dados sintéticos, dados de páginas da Web, dados de mídia social e dados de tipo misto. A pesquisa aborda muitos desafios que devem ser superados, bem como perspectivas futuras de pesquisa.

2.4 Considerações Finais

Colocando esses estudos em perspectiva, embora eles comparem diferentes abordagens e algoritmos de classificação, pode-se notar que eles tendem a identificar aplicações de rede por categoria (classificação de granulação grosseira) e não individualmente (classificação de granulação fina). Além disso, eles pretendem classificar o tráfego principalmente para fins de segurança e monitoramento, e sua complexidade computacional, que é medida pelo tempo gasto em treinamento e classificação, geralmente não é abordada.

Um ponto importande a se destacar, é que as características dos fluxo de tráfego podem ser divididas em duas classes de acordo com seus níveis de observação: (1) características de nível de fluxo; e (2) características de nível de pacote.

As características de nível de fluxo geralmente são calculados após a conclusão do fluxo, como número de pacotes, duração do fluxo e tamanho médio do pacote de um fluxo. Pelo contrário, as características de nível de pacote podem ser obtidas pela fase inicial do fluxo, tal como o comprimento do pacote, o tempo entre chegadas e a direção do pacote dos primeiros pacotes de fluxo. A intuição do recurso de nível de pacote é que os primeiros poucos pacotes constituem a fase de negociação do aplicativo, que são recursos descritivos.

Nos últimos anos, identificar qual a melhor classe a se utilizar para obter o melhor conjunto de características de um fluxo, tem se tornado foco profundo de pesquisas. Contudo, a literatura mostra que para um classificador de tráfego em tempo real obter um acurácia elevada, o mesmo deve tirar conclusões usando o menor número de pacotes possível (classe 2), em vez de esperar até que o fluxo seja concluído (classe 1) (Peng et al., 2015).

Capítulo 3

Redes Definidas por Software

3.1 Introdução

A classificação de tráfego tem sido amplamente utilizada no gerenciamento de redes, medições de serviços, projeto de redes, monitoramento de segurança e publicidade. As Redes Definidas por Software (*Software Defined Networks - SDN*) é uma tecnologia recém-desenvolvida, capaz de resolver problemas na rede tradicional, simplificando o gerenciamento da rede, introduzindo a capacidade de programação da rede e fornecendo uma visão global de uma rede.

Neste capítulo é apresentada a estrutura funcional da arquitetura do paradigma SDN e uma análise das pesquisas emergentes sobre técnicas de classificação de tráfego na SDN.

3.2 Redes Definidas por Software (SDN)

Redes Definidas por Software (*Software Defined Networks* ou SDN) é um paradigma de arquitetura de redes de computadores emergente, onde o controle da rede está dissociado do encaminhamento de pacotes e pode ser diretamente programável (FOUNDATION, 2012). Ou seja, a SDN apresenta como principal característica a separação do plano de controle do plano de encaminhamento, também conhecido como plano de dados, permitindo que o plano de controle possa ser movido do hardware de comutação (*switch* ou *roteador*) para um servidor dedicado. Dessa forma, o encaminhamento continua sendo eficiente, pois a consulta à tabela de encaminhamento continua sendo tarefa do hardware, mas a decisão sobre como cada pacote deve ser processado pode ser transferida para um nível superior, onde diferentes funcionalidades podem ser implementadas. Essa estrutura permite que a rede seja controlada de forma extensível através de aplicações, expressas em software. Para entender melhor os benefícios do

paradigma da arquitetura da SDN, na próxima seção é apresentada sua arquitetura padrão, assim como sua vantagem sobre o paradigma das rede atuais.

3.2.1 Arquitetura SDN

Em uma rede tradicional, a arquitetura de um comutador (*switches* e roteadores) é composta por duas camadas distintas: o software de controle, também denominado de plano de controle, e o *hardware* dedicado ao encaminhamento de pacotes, também denominado de plano de dados. Um *switch*, por exemplo, é construído com um *hardware* e um *software* proprietários. A Figura 3.1 ilustra um exemplo de uma rede tradicional com seus componentes.

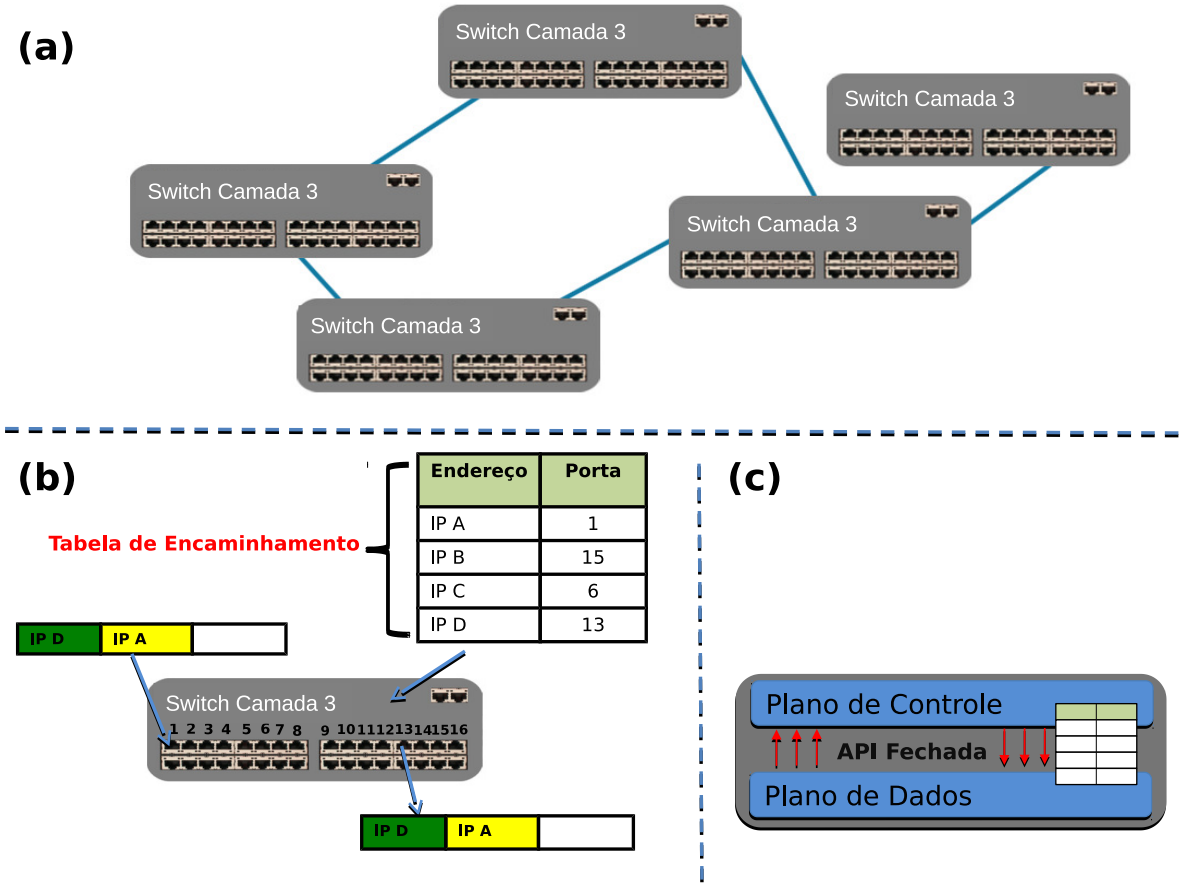


Figura 3.1: Exemplo de uma rede tradicional. (a) estrutura; (b) operação do comutador (switch); (c) organização interna do comutador (switch).

A primeira camada de um comutador da rede é o plano de controle ilustrado na Figura 3.1 (c), encarregado de tomar as decisões de encaminhamento e definindo a tabela de encaminhamento. Ele transfere a tabela de encaminhamento, por meio de

uma API (*Aplication Programming Interface*) proprietária, para a camada de plano de dados, Figura 3.1 (c), que realiza o encaminhamento dos pacotes ao nível de hardware, Figura 3.1 (b). A única interação do profissional responsável pela gerência da rede com o comutador é por meio de uma interface de configuração (*Web* ou *CLI*), limitando-se ao uso de funcionalidades básicas programadas pelo fabricante. Essas características implicam em ter muitas funções complexas embutidas na infraestrutura dos comutadores atuais, como gerenciamento, roteamento, firewall, controle de acesso, VPN, etc. Isso torna os comutadores uma arquitetura fechada, ou seja, com desenvolvimento fechado (software proprietário), sem facilidades para inserção de novas funcionalidade (ossificada).

A implantação da arquitetura das Redes Definidas por Software (SDN) pode resolver vários problemas no gerenciamento de rede da arquitetura atual. Essa nova arquitetura permite que a rede seja controlada de forma extensível através de aplicações, expressas em *software* alocados em servidores (chamados de controlador), e que o plano de dados continue alocado nos comutadores da rede (*switches* e roteadores). Ou seja, retira do comutador a lógica do plano de controle, transferindo essa funcionalidade para um controlador da rede.

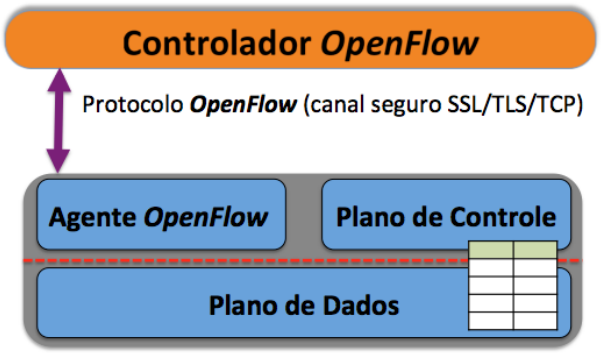


Figura 3.2: Arquitetura dos comutadores de rede proposto pela SDN com protocolo OpenFlow.

Na SDN, os comutadores de rede devem estar habilitados com um protocolo aberto para programação da lógica do equipamento, por exemplo OpenFlow (McKeown et al., 2008) (na prática um agente OpenFlow), para que haja um interfaceamento de comunicação entre ele e o controlador através de um canal seguro. No comutador da rede continuam existindo as duas camadas distintas (plano de controle e plano de dados). Contudo, a partir desta nova implementação, estes comutadores passarão a utilizar somente a camada do plano de dados, já que a camada do plano de controle será diretamente programável através de um servidor dedicado, denominado de controlador

OpenFlow, o qual definirá todas as regras do plano de controle, ou seja, definirá as tabelas de encaminhamento. A Figura 3.2 ilustra essa arquitetura que foi proposta pela primeira vez por (McKeown et al., 2010).

Com base nessa arquitetura de comutador, pode-se dizer que uma Rede Definida por Software (SDN) é caracterizada pela existência de um sistema de controle (software) dedicado em um servidor (controlador SDN), que pode controlar o mecanismo de encaminhamento dos elementos de comutação da rede por uma interface de programação bem definida (Guedes et al., 2012). Então, em uma arquitetura SDN a inteligência da rede é centralizada em controladores baseados em software SDN, que mantém uma visão global da rede, podendo abstrair a infraestrutura da rede para o desenvolvimento de novas aplicações, separando o plano de controle do plano de dados. A Figura 3.3 ilustra essa arquitetura.

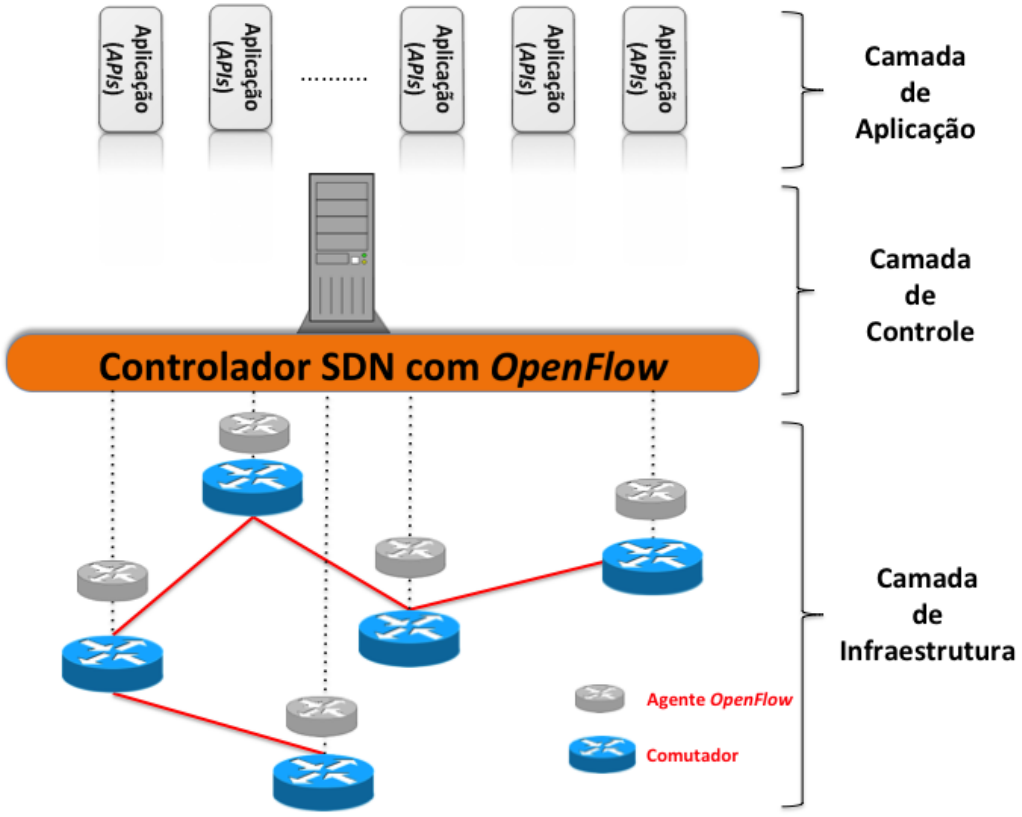


Figura 3.3: Arquitetura SDN.

A arquitetura SDN apresentada na Figura 3.3 é composta por três camadas. Na primeira camada, ou camada de infraestrutura, tem-se os comutadores da rede (*switch* ou roteador) com um agente do protocolo OpenFlow implementado. Esse agente é a interface de comunicação entre o controlador SDN e o comutador. Esses comutadores

possuem praticamente só o plano de dados e executam as ações e regras definidas pelo controlador SDN.

No segundo nível, camada de controle, tem-se o controlador baseado em *software*, implementado em um servidor dedicado. Este pode se comunicar com todos os elementos programáveis que compõem a rede SDN e oferecer uma visão unificada do estado da mesma, gerenciando assim as políticas de encaminhamento de pacotes. Um dos pontos fortes das SDN é exatamente a formação dessa visão unificada do estado da rede, sobre a qual é possível desenvolver análises detalhadas e chegar a decisões operacionais sobre como o sistema como um todo deve operar, simplificando assim a visão dos problemas e suas tomadas de decisões, bem com a implementação das ações a serem realizadas (Guedes et al., 2012).

Na camada superior, camada de aplicação, é possível desenvolver várias interfaces de programação (APIs) que permitem ao controlador oferecer diversos serviços de rede, incluindo a implementação de soluções de classificação de tráfego e de roteamento, especialmente customizadas para um ambiente particular, soluções de segurança, engenharia de tráfego para uma rede MPLS/GMPLS, QoS, esquemas de restauração de falhas para rede específicas como as redes ópticas elásticas, entre outros.

3.2.2 OpenFlow

De acordo com (McKeown et al., 2008) o protocolo OpenFlow é a primeira interface padrão de comunicação definida entre o controlador e as camadas de encaminhamento de uma arquitetura de Redes Definidas por Software. A Open Networking Foundation (ONF) (Committee et al., 2012) é responsável pela padronização do protocolo OpenFlow, a fim de garantir a interoperabilidade entre dispositivos de rede e software de controle de diferentes fornecedores. OpenFlow já está sendo amplamente adotado por fornecedores de infraestrutura.

O protocolo OpenFlow é implementado nos dois lados, entre a camada de infraestrutura de rede e a camada de controle SDN. O OpenFlow aplica o conceito de fluxos para identificar o tráfego da rede baseado em regras pré-definidas, dando a liberdade ao controlador SDN de programá-las de maneira estática ou dinâmica. Para tal, especifica um conjunto de instruções básicas que podem ser usadas para programar o plano de encaminhamento de cada dispositivo de redes. A Figura 3.4 ilustra algumas dessas instruções. O roteamento atual baseado em IP não fornece esse nível de controle, pois todos os fluxos entre dois pontos de extremidade devem seguir o mesmo caminho pela rede, independentemente de seus requisitos diferentes.

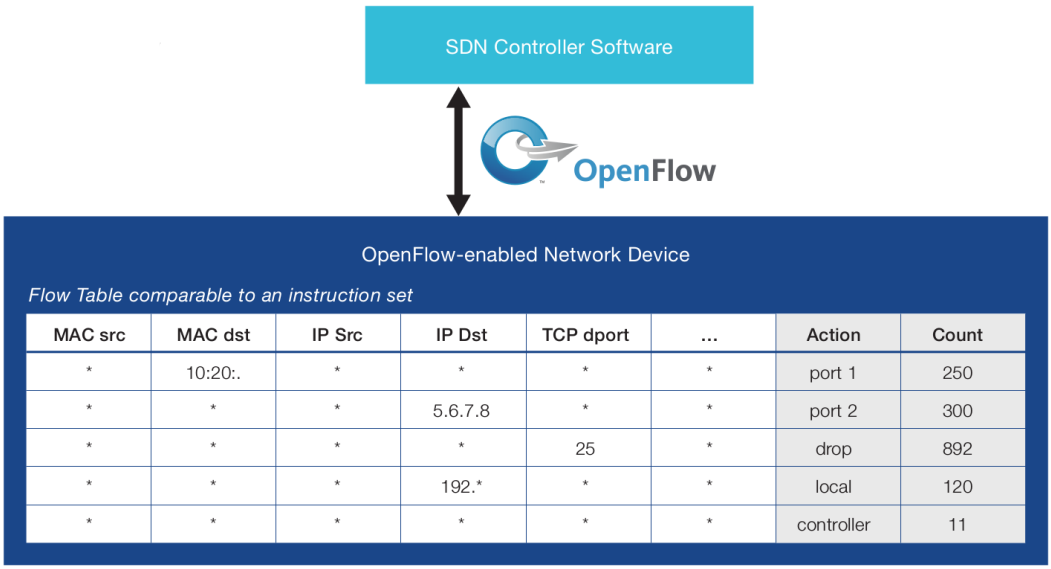


Figura 3.4: Conjunto de instruções do protocolo OpenFlow (Committee et al., 2012).

Uma rede programável com OpenFlow (Rothenberg et al., 2010) consiste em equipamentos de rede habilitados para que o estado da tabela de fluxos possa ser instalado através de um canal seguro, conforme as decisões de um controlador SDN:

- **Tabela de fluxos** - cada entrada na tabela de fluxos do hardware de rede consiste em regras, ações e contadores.
- **Canal seguro** - ele garante a confiabilidade na troca de informações entre o equipamento de rede e o controlador.
- **Protocolo OpenFlow** - um protocolo aberto para troca de mensagens entre os controladores e os equipamentos de redes.
- **Controlador** - é o software responsável por tomar decisões, adicionar ou remover entradas na tabela de fluxos, de acordo com o objetivo desejado.

O protocolo OpenFlow encontra-se em sua versão 1.5. Neste trabalho usaremos a versão 1.0.0 do protocolo, por se mostrar uma versão mais madura e avaliada nos trabalhos publicados de Redes Definidas por Software.

Dentro do protocolo OpenFlow os switches são chamados de *datapaths*. A tabela de fluxos é onde estão contidas as informações que formam os fluxos em que os pacotes devem ser encaminhados. Além disto ela contém também os contadores e as informações estatísticas a serem executadas e/ou mantidas por um *datapath*. Todos os

pacotes que passam por dentro de um *datapath* possuem seu cabeçalho confrontado com a tabela de fluxos.

Na especificação do OpenFlow 1.0.0 (Pfaff et al., 2012) um fluxo é formado pelos campos do cabeçalho, contadores e ações, conforme apresenta a Figura 3.5 e que são descritos como:

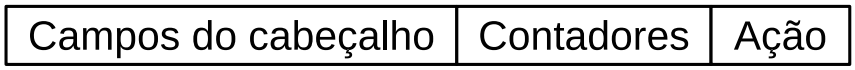


Figura 3.5: Campos de um fluxo OpenFlow (Pfaff et al., 2012).

- **Campos do cabeçalho:** informações encontradas nos cabeçalhos do pacote, porta de entrada e metadados usados para validar os pacotes de entradas.
- **Contadores:** usados na coleta de estatísticas para um fluxo em particular, como por exemplo, o número de pacotes de um fluxo entrante no *datapath*.
- **Ação:** que são aplicadas depois de cada validação as quais determinam como lidar com os pacotes correspondentes.

O campos do cabeçalho são verificados durante a comparação entre o pacote de entrada e tabela de fluxos. A Figura 3.6, apresenta os campos que constituem o cabeçalho de um pacote OpenFlow na versão 1.0.0.

Ingress Port	Ethernet src	Ethernet dst	Ethernet type	Vlan ID	Vlan Priority	IP src	IP dst	IP proto	IP TOS	TCP/UDP src port	TCP/UDP dst port

Figura 3.6: Campos do cabeçalho de um pacote OpenFlow (Pfaff et al., 2012).

Os contadores são mantidos por tabela, por fluxo ou por fila. Os contadores podem ser utilizados para a geração de estatísticas da rede. Existem várias aplicações *APIs* que consultam essas estatísticas a fim de monitorar a rede.

Os *datapaths* podem realizar diversas ações com os pacotes correspondentes a um determinado fluxo, sendo que um mesmo pacote pode corresponder a mais de um fluxo, portanto, a mais de uma ação diferente.

3.2.3 Cenários de Aplicação

Pela flexibilidade que as Redes Definidas por Software oferecem para a estruturação de sistemas em rede, o princípio estruturador oferecido por esse paradigma pode ser útil em praticamente todas as áreas de aplicação de Redes de Computadores. O controle da rede de forma programável oferecido pela SDN, combinado com outras tecnologias, pode ser aplicado na configuração, desempenho e no incentivo à pesquisa de novas aplicações nas operações de rede. A seguir, serão apresentados alguns trabalhos de aplicações para os quais soluções SDN foram propostas ou implementadas na literatura, a partir das perspectivas de aplicações de gerenciamento de rede, redes de Data Center e redes ópticas.

3.2.3.1 Gerenciamento de Rede

Para um gerenciamento eficiente de rede, são necessárias informações sobre o estado atual da rede. Em (Agarwal et al., 2013) são utilizadas informações da rede (coletados pelo controlador centralizado) para melhorar a utilização da rede e reduzir o atraso e as perdas de pacote. Em particular, o trabalho proposto se concentra no caso onde uma SDN é introduzida gradativamente em uma rede existente. Além disso, é formulado um problema de otimização do controlador SDN para engenharia de tráfego com implementação parcial. Para melhorar também os serviços de gerenciamento de redes, (Shirali-Shahreza e Ganjali, 2013) habilitou um controlador SDN OpenFlow para acessar informações em nível de pacote e tomar decisões de gerência. Em (Sharma et al., 2013) os autores combinam funções de gerenciamento das redes legadas (por exemplo, descoberta e detecção de falhas) com o provisionamento e controle de fluxo fim-a-fim habilitado pela SDN. A arquitetura atual das redes *core* celulares é constituída de elementos do plano de dados centralizados que mantêm conexões de estado que são caros e inflexíveis. Aplicações baseadas em SDN têm sido propostas para essas redes *core* celulares com objetivo de aumentar a flexibilidade e capacidade de gerenciamento das mesmas. No gerenciamento destas redes, o trabalho em (Nagaraj e Katti, 2014) propõe um novo projeto que elimina a manipulação desnecessária de fluxos em rede *core* LTE.

3.2.3.2 Redes de Data Center

Desde pequenas empresas a provedoras de computação em nuvem em grande escala, a maioria de seus sistemas e serviços de TI existentes são fortemente dependentes de Data Centers altamente eficientes e escaláveis. Contudo, suas infraestruturas ainda

representam desafios significativos em relação a custo, armazenamento e rede. Algumas pesquisas iniciais, de fato, mostram que as redes de Data Centers podem se beneficiar significativamente com o paradigma SDN na resolução de diferentes problemas. Um exemplo prático de uma aplicação real da arquitetura e concepção de uma SDN no contexto de Data Centers foi apresentado pela Google no início de 2012. A Google detalha em (Inter-Datacenter, 2012) uma implementação em larga escala de uma rede baseada em SDN conectando seus Data Centers. Em (Jain et al., 2013) é apresentado o projeto, implementação e avaliação do B4, uma WAN conectando mundialmente os Data Centers da Google. O trabalho explora o protocolo OpenFlow para conectar Data Centers da Google, buscando satisfazer os requisitos de largura de banda maciça, maximizar a largura de banda média e habilitar a medição e limitação de taxa na borda. Ao explorar o controle centralizado da SDN, o trabalho em (Hong et al., 2013) apresenta um sistema denominado de SWAN, que incrementa a utilização de redes inter Data Centers. SWAN controla quando e quanto de tráfego cada serviço envia e reconfigura o plano de dados para coincidir com a demanda do tráfego atual. Em (Krishnamurthy et al., 2014) é proposta uma nova abordagem para atribuição de *switches* SDN e partições do estado de aplicativo a instâncias de controlador distribuído, pois as arquiteturas atuais de Data Centers que utilizam controladores distribuídos não resolvem adequadamente dois grandes problemas: minimizar o tempo de latência de configuração do fluxo e minimizar o custo de operações do controlador através de alocação eficiente de recursos. Em (Zhu et al., 2015) é proposto um *timeout* inteligente para o protocolo OpenFlow, que atribui um *timeout* adequado a diferentes fluxos de acordo com suas características, bem como realiza um controle de *feedback* para ajustar o valor máximo do *timeout* de acordo com a atual ocupação da tabela de fluxo, com intuito de evitar *overflow* na tabela. Na proposta é adicionado um módulo de *cache* no controlador SDN baseado no OpenFlow, para gravar a última vez que um fluxo expira e usar o algoritmo baseado em histórico para prever o valor do *timeout* adequado.

3.2.3.3 Redes Ópticas

A utilização do paradigma da SDN com o protocolo OpenFlow em redes de transporte óptico pode oferecer muitos benefícios, que incluem principalmente melhorar o controle da rede e a flexibilidade no gerenciamento (FOUNDATION, 2013). Os potenciais benefícios e os desafios de extensão dos conceitos da SDN para várias arquiteturas de rede de transporte óptico (incluindo comprimento de onda óptico e comutadores de circuito e de fibra) são discutidos em (Gringeri et al., 2013). Em (Channegowda et al., 2013) é proposto um plano de controle baseado no paradigma SDN com o protocolo

OpenFlow para permitir operações em uma rede óptica tradicional, discutindo requisitos específicos e descrevendo a implementação de extensões OpenFlow para suportar redes de transporte óptico. Já em (Zhang et al., 2013d) é proposto um plano de controle baseado no paradigma SDN com o protocolo OpenFlow para redes ópticas elásticas. Em (Cvijetic et al., 2014) é projetado um acesso óptico dinâmico de grade flexível e agregação de rede. SDN é usado para controlar os lasers sintonizáveis no OLT (*Optical Line Terminal*) para transmissão de *downstream* flexível. Os comutadores seletivos de espectro (*SSS*) são controlados por meio de um controlador SDN para dinamicamente ajustar a banda passante (*passband*) para as transmissões *upstream* que chegam na OLT. (Yu et al., 2014) introduzem o conceito de *Spectrum Engineering*, que é utilizado para maximizar a eficiência espectral em redes ópticas com grade flexível que interconectam Data Centers. A desfragmentação de espectro, como um dos aspectos mais importantes do conceito de *Spectrum Engineering*, é demonstrado pela utilização do protocolo OpenFlow em redes ópticas de grade flexível. No contexto de restauração e recuperação de falhas, alguns trabalhos integrando a SDN foram propostos. Em (Savas et al., 2016) um esquema de reaprovisionamento de *backup* com proteção parcial de caminho (*BRPP*) baseado no plano de controle da SDN para redes ópticas WDM é proposto. Os recursos de backup são reservados, mas não provisionados (como na proteção de caminho compartilhado), de tal modo que a quantidade de largura de banda reservada para backups e seus roteamentos são sujeitos a mudanças dinâmicas, dado o estado da rede, aumentado assim a taxa de utilização da rede. Em (Aguado et al., 2016) é desenvolvido um mecanismo de recuperação de falha com reconfigurações virtuais dinâmicas utilizando o paradigma da SDN. O mecanismo considera *hypervisors* multi-domínios e controladores SDN de domínio específico para virtualizar as redes multi-domínios.

Outros trabalhos nas mesmas perspectivas dos citados acima, e também em outras perspectivas tais como Engenharia de Tráfego, *Middlebox*, segurança, virtualização de rede, mobilidade e redes sem fio, monitoramento e medição, entre outros, podem ser encontrados em (Nunes et al., 2014; Kreutz et al., 2015; Farhady et al., 2015).

3.3 Classificação de Tráfego em SDN

Nos últimos anos, a SDN trouxe novas oportunidades para classificar o tráfego em tempo real e realizar a seleção de suas características. A classificação precisa do tráfego no ambiente SDN é de fundamental importância para inúmeras outras atividades de rede, desde o monitoramento da segurança até a contabilidade, da Qualidade de Serviço (QoS) até o fornecimento de previsões úteis para o provisionamento de longo prazo.

Cada vez mais pesquisadores estudam a classificação de tráfego na SDN. Em (Zhang et al., 2013c) os autores propõem um método que utiliza propriedades dos fluxos, reunidas em um *dataset*, como informações de entrada do algoritmo K-means na fase de aprendizado, para *clustering*. Esses clusters são usados para implementar um modelo de classificação de tráfego. Embora a taxa de acurácia desse método seja de 89%, reduz a necessidade de investigar o conteúdo do pacote e o diagnóstico preciso de pacotes criptografados. Com o objetivo de ter um mecanismo de controle e decisão para obter o melhor desempenho, avaliando os parâmetros de QoS, (Arslan et al., 2013) propõem um controlador SDN baseado no simulador de eventos discretos Network Simulator 2 (ns2). Esse controlador classifica o tráfego de uma rede sem fio em dois tipos: com Taxa de Bits Constantes (Constant Bit Rate - CBR), baseado em tráfego *online*, e com Protocolo de Transferência de Arquivo (File Transfer Protocol - FTP), baseado em tráfego *offline*. Em (Qazi et al., 2013), os autores propõem um *framework* denominado de **Atlas** para classificação de tráfego em rede sem fio aplicando a SDN em ambiente com sistema operacional *Android*. No framework, foi adotado o algoritmo de Árvore de Decisão C5.0 para a classificação. O *framework* obtêm uma acurácia média de 94%, mas o mesmo é testado com apenas 40 aplicativos. No mundo real, há muito mais aplicativos móveis e fluxos de PC, o que pode degradar em muito a acurácia do método. Em (Jarschel et al., 2013) é aplicada a técnica de DPI em uma estrutura SDN para reconhecimento do fluxo, especificamente o streaming YouTube, possibilitando uma gerência do tráfego mais eficiente para melhorar a Qualidade de Experiência (QoE) do usuário. A estratégia adotada é a de reconhecer o fluxo de vídeo e direcionar o mesmo para um enlace reservado com suas características de QoS. Assim a largura de banda do fluxo do YouTube é garantida e uma boa QoE é fornecida. Os autores em (Li e Li, 2014) propõem uma abordagem, denominada de *MultiClassifier* para classificação de tráfego na camada de aplicação, combinando técnicas de DPI e algoritmo ML. O objetivo da abordagem é aproveitar as características das duas técnicas, para manter uma taxa de acurácia satisfatória e, ao mesmo tempo, atingir uma alta velocidade de classificação. O *MultiClassifier* obteve uma acurácia média de 85%. Contudo, os autores não apontam as características e o algoritmo ML utilizados e não mencionam os aplicativos contidos no conjunto de dados experimentais. (Cui et al., 2014) propõem um *framework* heurístico para implementar em redes SDN com OpenFlow a capacidade de classificação de tráfego baseado em DPI, para distinguir protocolos da camada de aplicação em tempo real. Seu objetivo é permitir que o controlador diferencie e isole diferentes fluxos de aplicativos, gerenciando e programando fluxos para garantir QoS para aplicativos sensíveis a atrasos. O trabalho proposto por (Da Silva et al., 2015) apresenta um *framework* para identificar, estender e selecionar conjuntos

de características de fluxo para classificação de tráfego de redes baseadas no protocolo OpenFlow. A coleta é obtida através de contadores nativos do OpenFlow, bem como de estatísticas matemáticas. O framework utilizou duas técnicas de seleção de características, a Análise de Componente Principais (PCA) e o Algoritmo Genético (GA), e o algoritmo de ML usado no *framework* proposto é o SVM. Os tráfegos foram classificados em três cenários: com características de DDoS, com características do protocolo FTP e com características de streaming de vídeo. As acurácias resultantes da classificação para cada cenário foram 97,33%, 94% e 88,67%, respectivamente. Os autores em (Ng et al., 2015) relatam as experiências práticas e as lições aprendidas durante o desenvolvimento de uma plataforma de classificação de tráfego baseada em SDN em uma rede corporativa. A plataforma proposta combina vários classificadores, em nível de software e de hardware, tornando-o um multiclassificador. No trabalho (Amaral et al., 2016) é proposto uma arquitetura para coletar dados de tráfego em uma rede corporativa usando o protocolo OpenFlow, que pode ser aplicada em redes SDN bem como em redes legadas. Os autores implementam três métodos de ML supervisionado em uma rede corporativa: Random Forests (RF), Stochastic Gradient Boosting (SGB) e Extreme Gradiente Boosting (EGB). Em (Wang et al., 2016) é implementado um esquema de classificação de tráfego com reconhecimento de QoS em SDN. A abordagem adotada estrutura o tráfego em classes, de acordo os diferentes requisitos de QoS das aplicações. O esquema combina a técnica de DPI para a rotulagem dos fluxos com um algoritmo semi-supervisionado Laplacian SVM para classificação das classes. Os fluxos foram categorizados em quatro classes de QoS: (1) Voice/Video Conference; (2) Interactive Data; (3) Streaming e (4) Bulk Data Transfer. Os autores em (Parsaei et al., 2017) propõem uma estrutura (*testbed*) para classificação de tráfego *on-line* utilizando o paradigma SDN. A classificação é baseada em características do fluxo do tráfego de acordo com cinco protocolos da camada de aplicação: FTP, HTTP, mensagens instantâneas, streaming de vídeo e peer-to-peer. Foram implementadas quatro variantes de Rede Neural para classificar o tráfego de acordo com o tipo de protocolo: feedforward, Multilayer Perceptron (MLP), NARX (Levenberg-Marquardt) e NARX (Naive Bayes). A estrutura proposta tem que coletar e calcular informações com base no fluxo total do tráfego. Estes quatro cenários de classificação obtêm precisão de 95,6%, 97%, 97% e 97,6%, respectivamente.

3.4 Considerações Finais

Neste capítulo foi apresentado o conceito fundamental do paradigma de Redes Definidas por Software (SDN), detalhando a sua arquitetura padrão, assim como as suas vantagens sobre o paradigma das redes atuais. Foram discutidos alguns trabalhos nos quais soluções SDN foram propostas ou implementadas, com ênfase para classificação de tráfego em rede.

É possível observar que a SDN é um paradigma de redes de computadores promissor para simplificar a gerência e controle da rede, possibilitando inovações de pesquisas e aplicações com base em seu paradigma de plano de controle centralizado, separado do plano de dados.

Contudo, com o rápido desenvolvimento das SDN, a classificação de tráfego obteve muitas conquistas, mas ainda há vários desafios a serem resolvidos. Para a classificação de tráfego em grande escala, os sistemas devem processar gigabits por segundo em muitos casos, o que é um grande desafio ainda. Outro aspecto é que as abordagens tradicionais tendem a classificar o tráfego com base no protocolo, denominada de classificação de granulação grossa, enquanto que mais e mais protocolos carregam diferentes aplicações. Assim, é cada vez mais importante classificar o tráfego de maneira refinada, pois visa distinguir a aplicação individual, fortalecendo o entendimento da rede e dos perfis de usuários, fornecendo uma melhor QoS. Além desses, o OpenFlow, a principal implementação de SDN atualmente, utiliza para a geração de características de fluxos de tráfego dois contadores nativos presentes em sua tabela de contadores, Byte Counter e Packet Counter, que não descrevem perfis de tráfego intrínsecos.

Capítulo 4

Método de Desenvolvimento do Módulo de Classificação de Tráfego

4.1 Introdução

Este capítulo apresenta o método utilizado para o desenvolvimento e a validação do módulo de classificação de tráfego de streaming de vídeo em tempo real com granularidade fina, baseado em aprendizado de máquina com uma nova abordagem para o relaxamento da hipótese de independência entre atributos do Naive Bayes.

4.2 Módulo de Classificação de Tráfego

O principal objetivo do presente trabalho foi o desenvolvimento de um classificador de tráfego em tempo real, capaz de identificar streamings de vídeo, para utilização em esquemas de Engenharia de Tráfego em Data Centers. Uma contribuição relevante foi obter a capacidade de lidar com multiclasse de tráfego de vídeo, considerando o nível de QoS necessário para cada aplicativo. O aspecto inovador foi o uso de um algoritmo ML baseado no modelo Naive Bayes, com uma nova abordagem quanto ao relaxamento da hipótese de independência entre atributos. Esta seção detalha o algoritmo de classificação e a arquitetura adotada para a implementação do módulo proposto.

4.2.1 Problema de Classificação

Conforme discutido no Capítulo 1, o tráfego de vídeo desempenha um papel cada vez mais importante no tráfego geral da rede. Um dos aspectos que torna o tráfego de vídeo um desafio, tanto para provedores de conteúdo quanto para infraestrutura de rede, é o requisito de largura de banda, como mostrado na Tabela 4.1. Em resposta a

esses requisitos, os principais provedores do mercado, YouTube e Netflix, têm investido em uma nova tecnologia chamada streaming adaptativo.

Tabela 4.1: Largura de banda recomendada para diferentes resoluções de vídeo (Netflix, 2019).

Resolução	Largura de Banda (Mbps)
SD	3.0
HD	5.0
Ultra HD	25.0

O streaming adaptativo consiste em uma técnica que, com base na detecção da largura de banda do usuário e da capacidade da CPU, ajusta a qualidade de um vídeo recebido para proporcionar a melhor experiência para o usuário. Para fazer isso, o streaming adaptativo determina quando solicitar novas partes de um vídeo e em qual resolução colocar essas solicitações ao tentar não extrapolar a capacidade da rede. Atualmente, os dois provedores citados usam o padrão MPEG-DASH (Dynamic Adaptive Streaming sobre HTTP, ISO / IEC 23009-1) (Lederer, 2015). A principal vantagem do MPEG-DASH é que ele é um padrão aberto e internacional, muito bem suportado pela indústria e disponível em muitas plataformas diferentes.

Considerando a disponibilidade do esquema de streaming adaptativo, a capacidade de espremer (comprimir) a taxa de transmissão de vídeo, as taxas exigidas por diferentes resoluções de vídeo (Tabela 4.1) e a natureza do serviço como gratuita ou uma assinatura, foram aqui propostas três classes de tráfego de vídeo, de acordo com os requisitos de prioridade de QoS para Engenharia de Tráfego. Essas classes são descritas na Tabela 4.2.

Tabela 4.2: Classes de Vídeo.

Classe	Características
A	Tráfego com maior prioridade. Em caso de falha, a restauração deve fornecer uma largura de banda garantida.
B	Tráfego com prioridade menor que a da Classe A. O tráfego que é restaurado após uma falha pode ser espremido. Ou seja, a recuperação garantida da largura de banda parcial é até o mínimo aceitável de QoS.
C	Tráfego com a menor prioridade em comparação com aqueles da Classe A e B. O tráfego restaurado após a falha pode usar recuperação de melhor esforço sem garantia de restauração.

Vídeos com requisitos de taxa mais altos e um requisito de assinatura, como Netflix, pertencem à Classe A. Os vídeos que usam streaming adaptativo e não têm assinatura, como o YouTube, estão incluídos na Classe B. O tráfego restante constitui a Classe C.

4.2.2 Algoritmo Proposto baseado no Classificador Naive Bayes

O classificador Naive Bayes baseado em uma distribuição gaussiana é um classificador probabilístico de baixa complexidade que segue o Teorema de *Bayes* (John e Langley, 1995). Se P(C) é a probabilidade do evento C ocorrer e P(C |X) é a probabilidade do evento C, dado que X ocorreu, então, do Teorema de Bayes, o seguinte é válido:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{4.1}$$

Esta regra é a base do classificador Naive Bayes (Langley et al., 1992). O classificador é chamado de '*naive*' porque assume a independência dos atributos. A suposição de independência significa que, para cada exemplo, o valor de um atributo não terá relação com o valor de nenhum outro atributo. Apesar do fato de que a independência dos atributos de uma instância é irreal, o classificador Naive Bayes é surpreendentemente eficaz na prática, uma vez que sua decisão de classificação pode estar correta, mesmo que suas estimativas de probabilidade sejam imprecisas.

No entanto, o trabalho apresentado em (Rish, 2001) analisa duas características que afetam o desempenho de um classificador Naive Bayes: (*i*) o impacto da entropia de distribuição sobre o erro de classificação e (*ii*) as dependências funcionais entre as características. A análise do item (*ii*) mostrou que o Naive Bayes funciona bem para certas dependências de características quase funcionais, alcançando assim seu melhor desempenho em dois casos opostos: *(a)* características completamente independentes e *(b)* características funcionalmente dependentes. Este trabalho aplica uma técnica que otimiza a característica do caso *(b)*, melhorando assim a Acurácia da classificação para o nosso problema.

Nesse sentido, a ideia central do Algoritmo Proposto (AP) é modificar o Naive Bayes de modo a não mais assumir a hipótese de independência entre os atributos. Em outras palavras, garante que a matriz de covariância seja positiva definida e simétrica. Como resultado, o Naive Bayes considera que a covariância entre os atributos é igual a zero (Langley et al., 1992).

A Figura 4.1 apresenta como o AP foi estruturado e o fluxo das chamadas das principais funções (função de treinamento, função estima PD e função de teste).

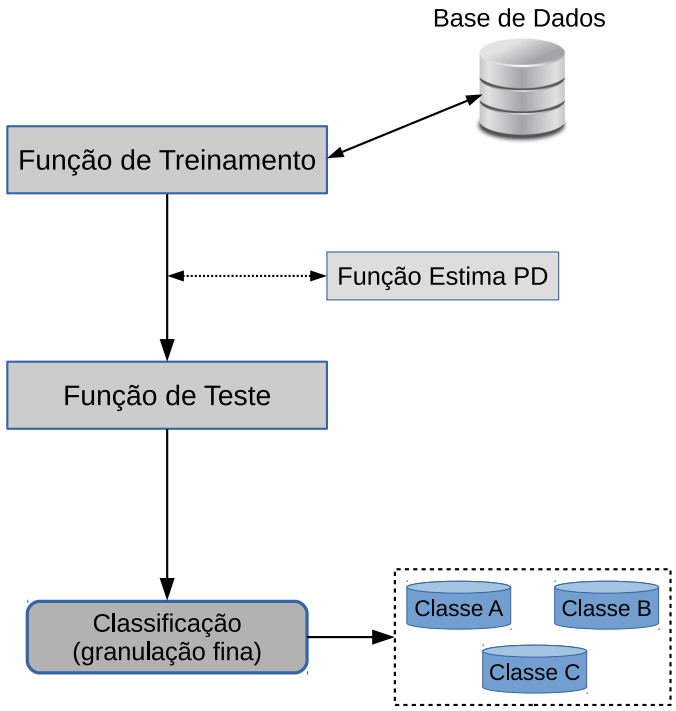


Figura 4.1: Fluxo funcional do Algoritmo Proposto.

A matriz de covariância desempenha um papel fundamental em muitas técnicas clássicas e modernas de análise de dados (Jolliffe, 2002) (Voyant et al., 2017) (Chen et al., 2017), particularmente quando se lida com um grande número de instâncias. A abordagem proposta foi adotada para garantir a não singularidade da matriz de covariância gerada para cada classe. Isso ajustou todos os autovalores para serem maiores do que alguns limites pré-especificados e, assim, garantiu uma matriz positiva definida. Assim, relaxou a matriz de covariância, transformando sua superfície de separação circular em uma elíptica.

Para isso, foi aplicada uma técnica capaz de garantir a viabilidade matemática da execução do novo algoritmo. Essa técnica calcula a matriz de covariância irrestrita para cada classe e estima a matriz positiva definida para as matrizes de cada classe que foram calculadas anteriormente. A estimação da matriz positiva definida é realizada através do conceito de minimização da norma Frobenius da diferença (Golub e Van Loan, 2012), pois ela tem a propriedade desejada de ser uma matriz simétrica positiva definida. A estratégia do algoritmo, que é produzir o estimador de matriz positiva definida para a matriz de covariância de cada classe, é baseada na função definida em (D'Errico, 2013). Essa função salva as matrizes de covariância geradas pelo algoritmo para cada classe (argumento de entrada), transformando-as com as características necessárias das matrizes Simétrica Positiva Definida (Symmetric Positive Definite - SPD) (argumento

de saída). O teste final da função é verificar se a matriz retornada (argumento de saída) realmente possui as propriedades de uma matriz positiva definida. Se não, faz o ajuste aplicando a Decomposição de Cholesky (Gentle, 2012). Se a fatoração retornar um segundo argumento com um valor zero, a matriz retornada terá as características de um SPD. Essa técnica de fatoração é usada para minimizar o problema da ortogonalidade de sinais. Onde:

Dada A: n x n Simétrica Positiva Definida, o fator de Cholesky G: n x n tal que $A = GG^t$ é obtido através do Algoritmo 1.

Algoritmo 1: Decomposição de Cholesky

para $k = 1, ..., n$ **faça**
  $s = 0$
  para $i = 1,..., (k-1)$ **faça**
    $s = s + g_{ki}^2$
  fim
  $s = a_{kk} - s$
  se $s \leq 0$ **então**
    Parar. A matriz A não é positiva definida.
  fim
  $g_{kk} = \sqrt{s}$
  para $j = (k+1)$,..., n **faça**
    $s = 0$
    para $i = 1,..., (k-1)$ **faça**
      $s = s + g_{ji} * g_{ki}$
    fim
    $g_{jk} = \frac{(a_{jk} - s)}{g_{kk}}$
  fim
fim

Ao otimizar os parâmetros, deve-se assegurar que a matriz resultante seja positiva definida, pois a matriz resultante é uma matriz de covariância, contendo as covariáveis entre os atributos do conjunto de dados. Neste caso, cada vez que os parâmetros são otimizados, a matriz deve ser verificada quanto à sua definição positiva. Se não é positiva definida, tem que ser ajustada, para que a definição positiva seja satisfeita.

Em (Higham, 1988) os autores descrevem um método que produz uma matriz SPD mais próxima de uma matriz positiva não definida. Para a implementação do método

descrito em (Higham, 1988), no Algoritmo Proposto, foi utilizada a função nearestSPD, que retorna a matriz SPD mais próxima mencionada. A função foi aplicada para lidar com a violação de um autovalor negativo. Se houver uma violação, a matriz é atualizada usando a Decomposição de Cholesky (Gentle, 2012), que calcula seus autovalores.

O Algoritmo 2 apresenta os passos implementados na função de estimação da matriz simétrica positiva definida, descrita acima.

Algoritmo 2: *Função Estima SPD (nearestSPD)*

Entrada: Matriz_Cov
Saída: SPD
início
  Passo 1: Testa argumento de entrada
  Passo 2: **Se** Matriz_Cov é escalar e não positiva **Então** Ajusta
  Passo 3: Gera a matriz simétrica para Matriz_Cov
  Passo 4: Testa se matriz gerada no **Passo 3** possui características de SPD
  Passo 5: **Se não** for SPD
  Ajusta aplicando a decomposição de **Cholesky** `// Algoritmo 1`
  Passo 6: Salva a SPD
fim

Nesse sentido, um dos fatores que contribui para um melhor desempenho na Acurácia do Algoritmo Proposto é o ajuste que a função faz na matriz de covariância, de modo que a independência entre os atributos não mais exista. De fato, aplicando-se ao Naive Bayes Gaussiano, a técnica de seleção de características proposta neste trabalho (Seção 4.3.2) produziu uma melhoria significativa no desempenho, em torno da mesma ordem de grandeza do Algoritmo Proposto, sem aplicar a seleção de características. Dessa forma, o Algoritmo Proposto realiza essa seleção automaticamente, ajustando a matriz de covariância utilizada pelo Naive Bayes Gaussiano. Uma das razões pelas quais foi utilizada uma comparação com e sem seleção de características foi justamente para verificar essa premissa.

Neste trabalho, os códigos das funções de treinamento e teste são descritos de tal forma que é possível adotar qualquer função de normalização desejável, qualquer tamanho de conjunto de dados desejável (a porcentagem do conjunto de dados usado em treinamento e teste) e qualquer métrica de avaliação de desempenho desejável. Nessa etapa, os códigos foram implementados usando a linguagem de programação M-code da ferramenta Matlab R2016a (MATLAB, 2016).

O Algoritmo 3 mostra a função de treinamento implementada.

Algoritmo 3: *Função de Treinamento*

```
1  function [training_model] = newgaussianBayes(x,y)
2  xA = x(y==0,:); xB = x(y==1,:); xC = x(y==2,:);
3  [n,~] = size(x);
4  [nA,~] = size(xA); [nB,~] = size(xB); [nC,~] = size(xC);
5  muA = mean(xA); muB = mean(xB); muC = mean(xC);
6  SigmaA = cov(xA); SigmaB = cov(xB); SigmaC = cov(xC);
7  NSPDsigA = nearestSPD(SigmaA);
8  NSPDsigB = nearestSPD(SigmaB);
9  NSPDsigC = nearestSPD(SigmaC);
10 classA.mu = muA; classA.sig = NSPDsigA; classA.priori = nA/n;
11 classB.mu = muB; classB.sig = NSPDsigB; classB.priori = nB/n;
12 classC.mu = muC; classC.sig = NSPDsigC; classC.priori = nC/n;
13 training_model.cA = classA;
14 training_model.cB = classB;
15 training_model.cC = classC;
16 return
17 end
```

A linha 2 inicia as amostras de classe a serem usadas, onde **x** é uma matriz contendo os pacotes de fluxo de tráfego como linhas e suas 14 características (apresentadas na Tabela 4.3, exceto a variável de class) como colunas. Por sua vez, **y** é um vetor contendo o mesmo número de linhas que a matriz **x**, onde cada valor fornece a classe associada ao pacote na linha correspondente da matriz **x**. Os valores em **y** são codificados como: 0 corresponde à classe A, 1 à classe B e 2 à classe C, respectivamente.

A linha 3 define o tamanho da amostra para o conjunto de treinamento e, em seguida, a linha 4 define o tamanho da amostra para cada classe. Nas linhas a seguir, a mesma ação é executada em cada classe. Na linha 5, o vetor média é gerado. A linha 6 calcula a matriz de covariância irrestrita. As linhas 7, 8 e 9 chamam a função definida em (D'Errico, 2013), chamada the nearestSPD, que transforma cada matriz de covariância calculada na linha 6 em matrizes tipo Positiva Definida. Em sequência, as

linhas 10, 11 e 12 constroem o modelo probabilístico baseado no Teorema de Bayes para cada matriz Positiva Definida. Finalmente, as linhas 13, 14 e 15 executam o modelo de treinamento.

O Algoritmo 4 mostra a função de teste implementada.

Algoritmo 4: *Função de Teste*

```
1  function [test] = testnewgaussianBayes(x,y, training_model)

2  densA = mvnpdf(x, training_model.cA.mu,training_model.cA.sig);
3  densB = mvnpdf(x, training_model.cB.mu,training_model.cB.sig);
4  densC = mvnpdf(x, training_model.cC.mu,training_model.cC.sig);

5  postA = densA*training_model.cA.priori;
6  postB = densB*training_model.cB.priori;
7  postC = densC*training_model.cC.priori;

8  y_hat = zeros(1,length(y));
9  y_hat(postB>postA & postB>postC) = 1;
10 y_hat(postC>postB & postC>postA) = 2;

11 end
```

As linhas 2, 3 e 4 calculam a densidade de probabilidade da distribuição normal multivariada do ponto **x** para cada classe. Como tal, o Algoritmo 3 aplica a função **mvnpdf** do Matlab (função de densidade de probabilidade normal multivariada). As linhas 5, 6 e 7 executam o cálculo da probabilidade a posteriori, novamente para cada classe. A inferência da condição de classificação é calculada pelas linhas 8, 9 e 10.

4.2.3 Arquitetura do Classificador

Para implementar o módulo de classificação, foi definida uma arquitetura contendo quatro blocos funcionais (Figura 4.2): Captura, Pré-processamento, Treinamento e Validação de pacotes.

4.2.3.1 Bloco de Captura de Pacotes

Este módulo captura pacotes de rede usando uma placa de rede no modo promíscuo. O bloco extrai dados de tráfego da interface de rede e os salva em um arquivo para processamento adicional. O aplicativo que executa essa tarefa é chamado de *sniffer* e, aqui, Wireshark (Chappell e Combs, 2010) foi usado, mais especificamente sua ferra-

menta *dumpcap*. O *dumpcap* captura os dados de uma interface de rede, os salva em um arquivo **.pcap** e oferece recursos, incluindo filtros de captura, interfaces e condições de parada, como tempo. Para simplificar a implementação do protótipo, essa captura foi feita na mesma máquina que estava executando os aplicativos. Todos os pacotes foram capturados de uma conexão Wi-Fi 802.11n, com uma taxa de download limitada a 10 Mbps. Vale ressaltar que o módulo foi concebido para ser usado no nível de rede (nível IP). Assim, as características dos níveis mais baixos (níveis físicos e de enlace) não afetam o perfil de tráfego e o módulo é capaz de operar em qualquer infraestrutura, como Wi-Fi, FTTH, óptica ou outras. Além disso, o método de seleção de características usado no algoritmo fornece mais flexibilidade e permite ajustar a diferentes características de rede, bem como diferentes estados. Como resultado, o algoritmo se adapta melhor a mudanças na configuração da rede, bem como à circulação de mais ou menos quantidades de dados.

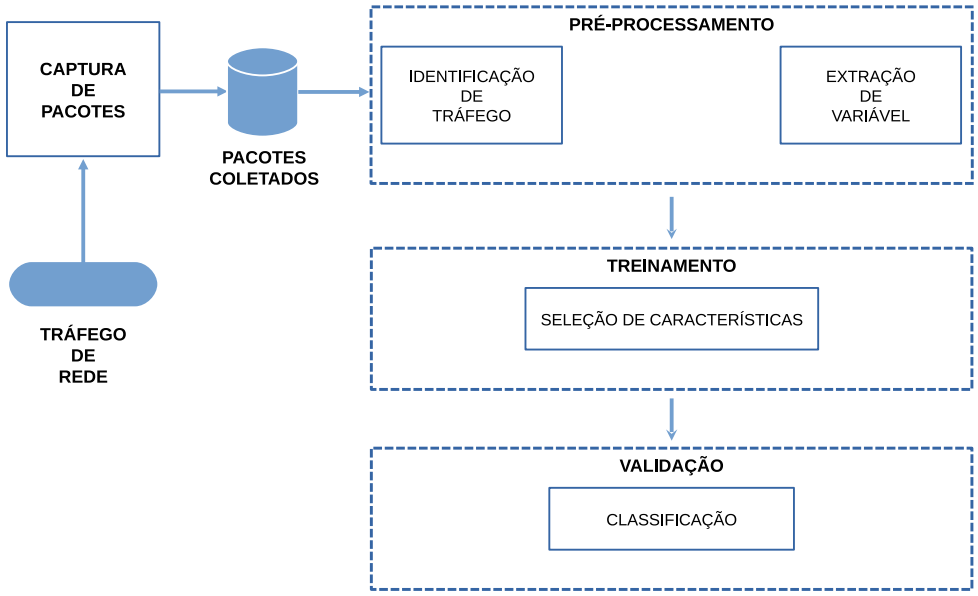


Figura 4.2: Arquitetura proposta para o módulo classificador.

4.2.3.2 Bloco de Pré-Processamento

O bloco de pré-processamento é responsável por receber um arquivo de captura no formato **.pcap** e gerar arquivos com as informações necessárias para os blocos a seguir. O bloco é composto por duas tarefas: Identificação de Tráfego e Extração de Variável. A tarefa de Identificação de Tráfego rotula cada pacote de acordo com o aplicativo de origem: Netflix, YouTube ou tráfego em segundo plano (background traffic). Isso é necessário porque o classificador usa uma abordagem supervisionada. A tarefa

de Extração de Variável calcula um conjunto de variáveis estatísticas que são todas derivadas dos cabeçalhos de pacotes e executa a extração de características. Apenas o tráfego IPv4 foi considerado neste trabalho, sendo que a Figura 4.3 ilustra os diferentes campos do cabeçalho dos pacotes. O suporte para tráfego IPv6 será adicionado em versões futuras.

Figura 4.3: Campos do cabeçalho IPv4 (Peterson e Davie, 2007).

Os pacotes capturados (**.pcap**) foram previamente analisados usando o analisador de pacotes Wireshark (Chappell e Combs, 2010). A análise forneceu informações críticas sobre os dados explorados ao extrair as características. O tráfego de streaming de vídeo apareceu com características em rajadas (*bursts*), com uma grande quantidade de pacotes de vídeo muito semelhantes chegando quase ao mesmo tempo. Em contraste, o tráfego em segundo plano foi mais irregular em características e horário de chegada (*arrival time*). Essas informações foram importantes para identificar e definir as características mais descritivas para o problema, dos campos do cabeçalho IP.

As características extraídas foram divididas em dois grupos. O primeiro grupo consiste nos parâmetros encontrados nos campos de cabeçalho IPv4. A partir dos campos mostrados na Figura 4.3, nem todos são úteis para a classificação. Por exemplo, o campo Checksum de Cabeçalho varia especificamente com cada pacote de rede e o campo Versão será sempre o mesmo (já que não consideramos o IPv6 neste trabalho). Para os experimentos, os campos mais relevantes foram Comprimento do Pacote, Flag, Deslocamento de Fragmento, Tempo de Vida e Protocolo, além do comprimento do quadro Ethernet que contém o pacote IP.

O segundo grupo de características extraídas consiste nos parâmetros calculados a partir de campos de mais de um pacote IP. Isto é necessário para verificar a natureza em rajadas do tráfego de streaming de vídeo. O *script* de extração de característica, desenvolvido, considera uma janela W de três pacotes e calcula todos os valores baseados nessa janela, exceto aqueles derivados do tempo de chegada entre dois pacotes consecutivos. Os valores são: o tempo entre os pacotes, a média e a variância do tempo entre os pacotes, a média e a variância do comprimento total do IP, a média e a variância do campo Tempo de Vida e a média e a variância do campo Protocolo.

No total, existem 14 características, além da classe de tráfego (identificação de tráfego). A Tabela 4.3 apresenta as variáveis extraídas com suas abreviações e descrições. Um *script* de extração desenvolvido em Python gera dois arquivos, um com os rótulos e outro com as características extraídas. Para executar a extração, o script usou a biblioteca **dpkt** (Song, 2015) com as definições básicas dos protocolos TCP/IP para examinar o conteúdo do pacote no arquivo **.pcap**.

Tabela 4.3: Abreviação e descrição de cada variável extraída.

Variável	Descrição
ipt	Tempo entre a chegada dos pacotes (segundos).
pa_s	Comprimento do quadro Ethernet (octetos).
ip_l	Comprimento do datagrama IP (octetos).
ip_of	Valor decimal referente a concatenação dos campos Flag e Deslocamento de Fragmento do datagrama IP.
ip_p	Valor decimal referente ao campo Protocolo do datagrama IP.
ip_ttl	Valor decimal referente ao campo Tempo de Vida do datagrama IP.
m_iat	Média do tempo entre chegada dos pacotes (segundos).
v_iat	Variância do tempo entre as chegada dos pacotes.
m_ip	Média do comprimento do datagrama IP.
v_ip	Variância do comprimento do datagrama IP.
m_ttl	Média do valor decimal referente ao campo Tempo de Vida do datagrama IP.
v_ttl	Variância do valor decimal referente ao campo Tempo de Vida do datagrama IP.
m_p	Média do valor decimal referente ao campo Protocolo do datagrama IP.
v_p	Variância do valor decimal referente ao campo Protocolo do datagrama IP.
class	Classificação da amostra.

4.2.3.3 Bloco de Treinamento

No bloco Treinamento a intenção, baseada em todos as características extraídas pelo bloco de Pré-processamento, é executar a seleção de características. Esta seleção permite construir um modelo flexível que não é necessariamente baseado nas mesmas características e oferece uma classificação mais fiel do conjunto de dados.

Para o método de seleção de características, adotamos a abordagem de remoção de variáveis que possuem um alto valor de correlação com outras variáveis, mas não possuem uma correlação apreciável com sua classe. Ao fazer isso, a ideia é remover as características que são redundantes para o problema, mantendo as características relevantes para a classificação. Isso pode ser feito estabelecendo um intervalo de corte

para os valores de correlação entre as variáveis de entrada e outro intervalo de corte para a correlação entre as variáveis de entrada e de classificação. Neste trabalho, as faixas adotadas foram **0,7** e **0,3**, respectivamente, seguindo (Hall, 1999) (Hall, 2000).

4.2.3.4 Bloco de Validação

O bloco de Validação tem como função validar o melhor classificador para o problema, antes de empregá-lo para classificar a rede. Essa validação usa parte dos dados extraídos da captura. Pretendeu-se obter um classificador que, para o tráfego de rede, produzisse a maior Acurácia, tivesse uma boa taxa de acertos por classe e respeitasse os limites de custo computacional. Ou seja, a validação obtém o melhor *trade-off* entre o desempenho e os custos computacionais da classificação.

4.3 Experimentos de Validação do Módulo

Estes experimentos foram desenvolvidos para analisar e avaliar o módulo para uso em cenários de classificação de tráfego de rede em tempo real, com um plano de controle centralizado (ambiente SDN). Um conjunto de dados (dataset) foi produzido com base no tráfego de rede capturado de três tipos diferentes de aplicativos. Uma vez gerado, o conjunto de dados foi utilizado para avaliar o comportamento de dois algoritmos de classificação ML: o Naive Bayes Gaussiano clássico e o algoritmo de ML proposto, que é composto pelos algoritmos newgaussianBayes e testnewgaussianBayes, como funções de treinamento e teste respectivamente.

4.3.1 Gerando o Conjunto de Dados

Uma das dificuldades foi encontrar, na literatura ou em repositórios de aprendizado de máquina (Lichman, 2015) (Kaggle, 2016), conjuntos de dados contendo vídeos em streaming do YouTube e Netflix com as características desejadas. As características necessárias eram específicas do tráfego, como as camadas de enlace e de rede, enquanto os conjuntos de dados disponíveis em repositórios forneciam apenas informações da camada de aplicativo (nível do navegador da web). Além disso, os dados disponíveis estavam desatualizados, já que foram produzidos em aproximadamente 2011, como em (Rao et al., 2011a). Dessa forma, a solução foi gerar um novo conjunto de dados com as características de tráfego de streaming de vídeo necessárias para esse trabalho.

Os scripts bash do Linux foram usados para criar o conjunto de dados considerando cada tipo de serviço acessado no experimento: streaming de vídeo do YouTube, stre-

aming de vídeo da Netflix e downloads de arquivos. A Listagem 4.1 mostra o script implementado para o serviço do YouTube.

Listing 4.1: *Shell Script* para o Serviço de *Streaming* de Vídeo.

```
#!/bin/bash
export DISPLAY=:0
echo $HOME > home.txt
source $HOME/.bashrc
export names=("sony4k", "costaRica", "nasa")
export videos=("oh904_HdkwY", "iNJdPyoqt8U", "fVMgnmi2D1w")
export n=$((RANDOM%3))
export timeSample=$(date +"%H-%M_%d-%m-%Y")
export name=$(echo ${names[2]} | sed 's/,$//')
export video=$(echo ${videos[2]} | sed 's/,$//')
google-chrome-stable https://www.youtube.com/embed/$video?autoplay=1 &
dumpcap -i wlp3s0 -a duration:120 -P -w \
/home/linux/PythonStuff/captures/youtube/output-$name-home\_$timeSample.pcap
cd /home/linux/PythonStuff/captures/youtube
python /home/klenilmar/featureExtractionv2.py -f &
/home/linux/PythonStuff/captures/youtube/output-$name-home\_$timeSample.pcap -t 1 -o $name
echo $n
pkill --oldest chrome
pkill chrome
echo "all done!"
```

Cada *script* começa acessando um endereço da web, usando uma janela do navegador *Google Chrome* para iniciar uma captura de tráfego de rede. No final da captura, os dados são salvos em um arquivo cujo nome contém um identificador de vídeo e a data de início da captura. Na sequência, um script Python lê o arquivo **.pcap** e gera os arquivos com as características extraídas e suas respectivas classes (tarefas de Identificação de Tráfego e de Extração de Variáveis). Vale ressaltar que o desenvolvimento desses scripts também levou em conta sua execução pelo **CRON**, um escalonador de tarefas amplamente utilizado em sistemas Unix (Linux, 2018). A ideia é que, em versões futuras, esses scripts possam ser executados periodicamente, tornando possível gerar um novo conjunto de dados e um novo modelo de classificação a cada hora. O experimento produziu uma captura de 120 segundos composta de três vídeos do YouTube, dois vídeos do Netflix e dois downloads de arquivos, usando scripts semelhantes ao da Listagem 4.1.

Os dados extraídos foram salvos em dois arquivos, um de características e um de rótulos, produzindo assim uma base cuja distribuição por classe é apresentada na Tabela 4.4.

Tabela 4.4: Distribuição do Conjunto de Dados para cada tipo de Aplicativo.

Classe	Tipo	Número de Pacotes
A	Netflix Video	278.687
B	YouTube Video	118.147
C	File Downloads	154.261
-	**Total**	**551.095**

4.3.2 Análise dos Dados

A análise do conjunto de dados começou calculando as estatísticas e distribuições para cada variável de entrada. Os resultados estão apresentados nas Tabelas 4.5 e 4.6 e nos gráficos das Figuras 4.4, 4.5 e 4.6. Nenhum gráfico foi plotado para as variáveis **v_iat** e **v_p**, pois todos os seus valores estavam próximos de zero.

Tabela 4.5: Estatísticas das Variáveis - Grupo 1.

	ipt	pa_s	ip_l	ip_of	ip_p	ip_ttl	m_iat
Mínimo	-0.001291	46	32	0	1000	1.0	-0.000383
25%	0.000018	80	66.0	6000	6000	61.0	0.00039
Mediana	0.000448	1514	1500	16384	600	61.0	0.000668
Média	0.001511	1009	995.1	16384	8306	59.58	0.001518
Desvio Padrão	0.0440047	665.227	665.227	16305	4.47774	7.609278	0.0329422
75%	0.001242	1514	1500.0	16384	6000	64.00	0.001015
Máximo	11.907075	1514	1500.0	16384	17000	255.00	7.666284

Tabela 4.6: Estatísticas das Variáveis - Grupo 2.

	v_iat	m_ip	v_ip	m_ttl	v_ttl	m_p	v_p
Mínimo	0	40	0	1	1000	4333	0
25%	0	1017.3	380774	53.33	2	6000	0
Mediana	0	1017.3	465934	62	2	6000	0
Média	0.0009	995.2	397874	59.58	26.27	8306	0.1173
Desvio Padrão	0.1007	211.3001	139432.8	5.625483	248.4432	4.464570	1.7875
75%	0	1017.3	465934	62	56.89	6000	0
Máximo	34.8368	1500	524176	237.67	12168.00	17000	56.8889

Outra observação interessante é que o tempo entre a chegada dos pacotes (**ipt**) e consequentemente sua média (**m_iat**) podem ser negativos. Isto é possível, devido a reordenação de pacotes e perda durante a transmissão.

Para cada gráfico, os valores das variáveis foram normalizados seguindo a Equação 4.2, garantindo assim que os dados pertencem ao intervalo [0,1]. **N** é o número de

observações usadas para fazer cada um dos gráficos (ou seja, **N** é o número de pacotes do conjunto de dados gerado). O ***k*** indica uma dimensão/coluna de uma observação $\mathbf{x}_i$ (não há relação aqui com o fator k de um procedimento de validação cruzada). O parâmetro Bandwidth que aparece nas figuras de distribuição de densidade indica o grau de suavização.

$$(x_i^k)_N = \frac{x_i^k - x_{min}^k}{x_{max}^k - x_{min}^k} \tag{4.2}$$

Os gráficos de densidade e informações estatísticas mostram que algumas variáveis não estão distribuídas uniformemente em seu intervalo e apresentam picos estreitos. Isso ocorre para as variáveis **ipt**, **ip_of**, **m_iat** e **v_ttl**, como na Figura 4.4.

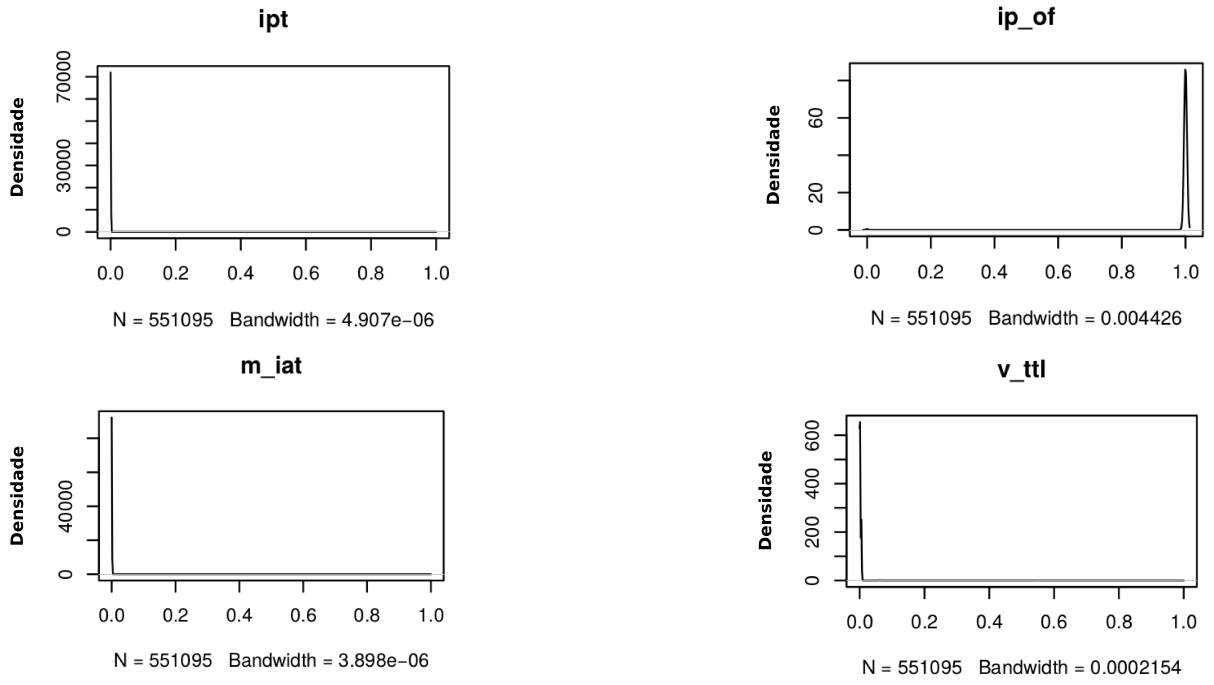


Figura 4.4: Gráficos de variáveis com picos na distribuição de densidade.

Adicionalmente, observa-se que as variáveis **pa_s** e **ip_l**, assim como **ip_p** e **m_p**, apresentaram perfis bimodais (Figura 4.5).

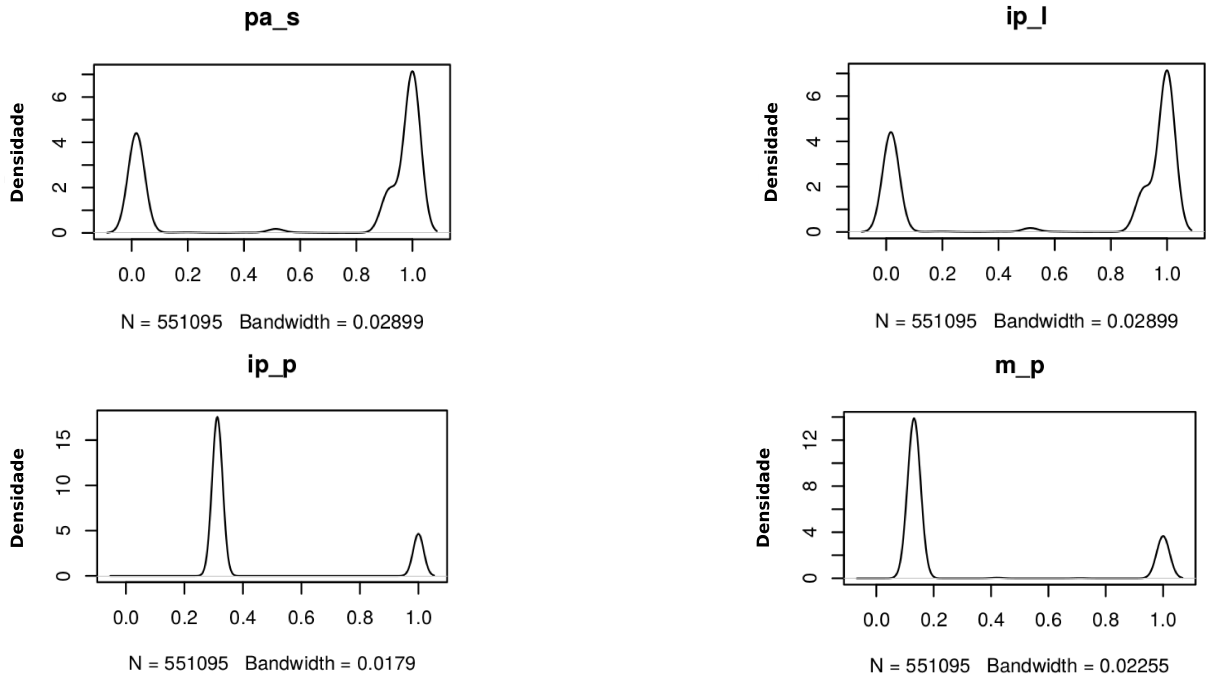


Figura 4.5: Gráficos de variáveis com distribuição de densidade bimodal.

Outras variáveis apresentaram uma distribuição multimodal, como **ip_ttl**, **m_ip**, **v_ip** e **m_ttl**, como mostra a Figura 4.6.

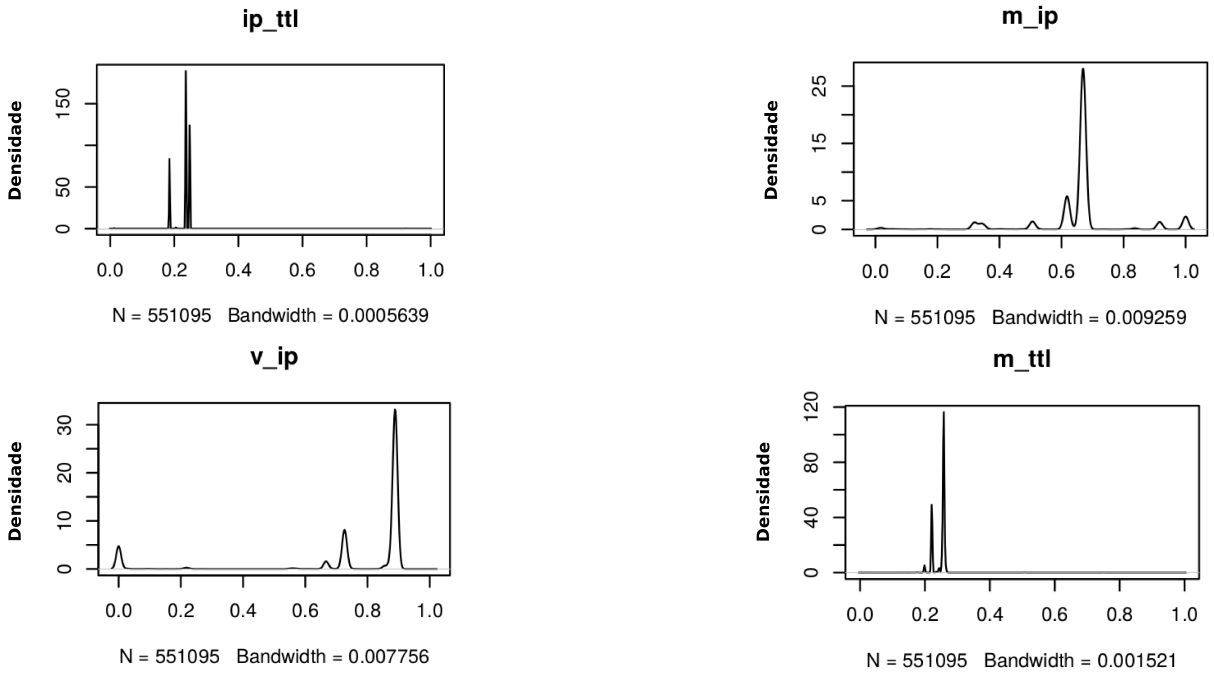


Figura 4.6: Gráficos de variáveis com distribuição de densidade multimodal.

Para verificar o impacto das características mais relevantes, o gráfico da Figura 4.7 foi gerado com as correlações entre as variáveis do problema, incluindo a variável de saída.

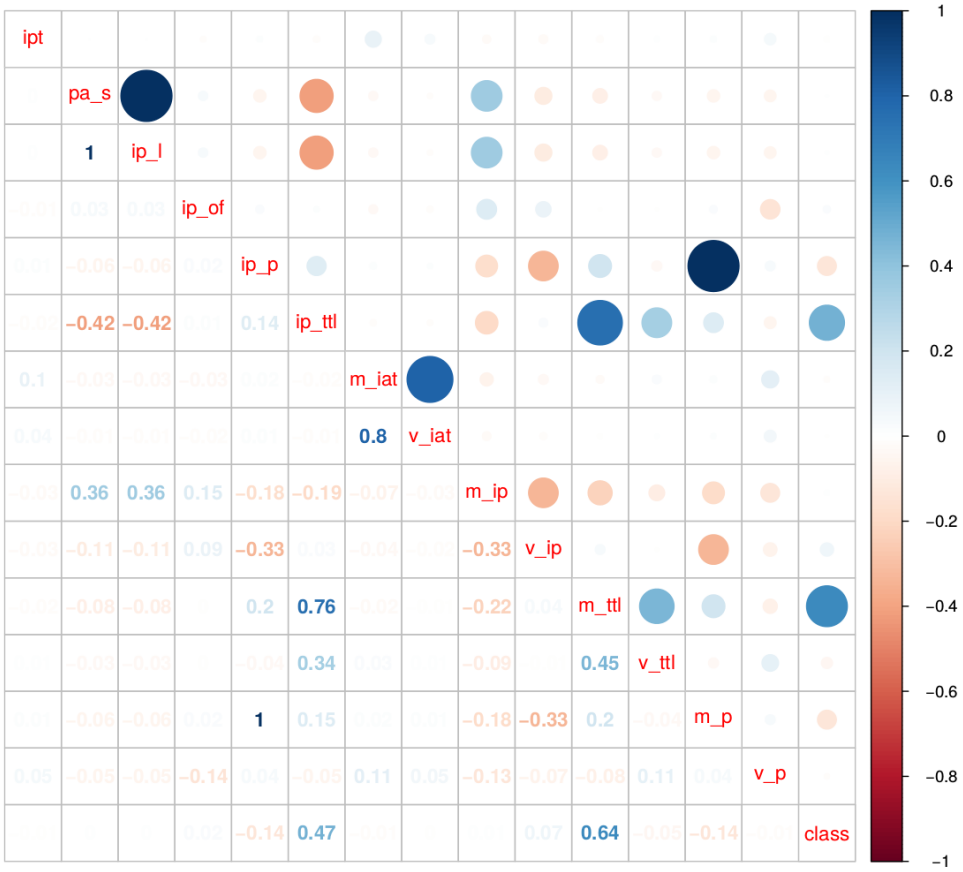


Figura 4.7: Gráfico de correlação entre as variáveis do problema.

Examinando o gráfico da Figura 4.7, observa-se que os pares de variáveis (**pa_s**, **ip_l**) e (**ip_p**, **m_p**) têm correlações iguais a um, o que significa que para cada par (**X**, **Y**), uma equação linear descreve a relação entre **X** e **Y** perfeitamente, com todos os pontos posicionados em uma linha na qual **Y** cresce à medida que **X** cresce. Por esta razão, concluímos que as características são redundantes. Além disso, os pares (**m_iat**, **v_iat**) e (**ip_ttl**, **m_ttl**) apresentaram valores de correlação considerados fortes, com coeficientes maiores que **0.7**, e poderiam ser removidos da base. No entanto, as variáveis do par (**ip_ttl**, **m_ttl**) tiveram correlações apreciáveis com a variável de classificação, o que indica que elas são características relevantes para o problema e sua remoção não é recomendada.

Em conclusão, a proposta foi remover três características do total de 14 e comparar o desempenho do classificador em termos de Acurácia e complexidade computacional. O objetivo era verificar se essa seleção de características levaria a benefícios em termos

do desempenho global do modelo. As três características removidas foram **pa_s**, **m_iat** e **m_p**.

4.3.3 Desempenho do Classificador

Existem várias maneiras de medir o desempenho de um modelo de aprendizado de máquina. As melhores métricas de previsão para um modelo de aprendizado de máquina são dependentes do cenário. Como a proposta deste trabalho está diretamente ligada à classificação de tráfego *on-line*, as métricas utilizadas devem expressar o melhor compromisso entre o desempenho geral e a complexidade computacional. Também é importante escolher um método de validação (Acurácia do classificador) para o classificador escolhido.

Para validar o desempenho da abordagem, foi utilizado o método de validação cruzada (*cross-validation*) (Chandrashekar e Sahin, 2014). A técnica de validação cruzada do tipo *k-fold* foi aplicada com k = 10. Na validação cruzada de *10-fold*, a amostra original foi dividida aleatoriamente em dez subamostras de tamanho igual. Uma única subamostra foi retida para teste, enquanto os nove restantes foram usados para treinamento. A Acurácia obtida para a subamostra do teste foi registrada. O processo foi repetido dez vezes e cada iteração usou uma subamostra diferente para teste. Os resultados foram então calculados para produzir uma única estimativa. A principal vantagem deste método é que todas as amostras foram usadas apenas uma vez para testes. A validação cruzada de *10-fold* é ilustrada na Figura 4.8.

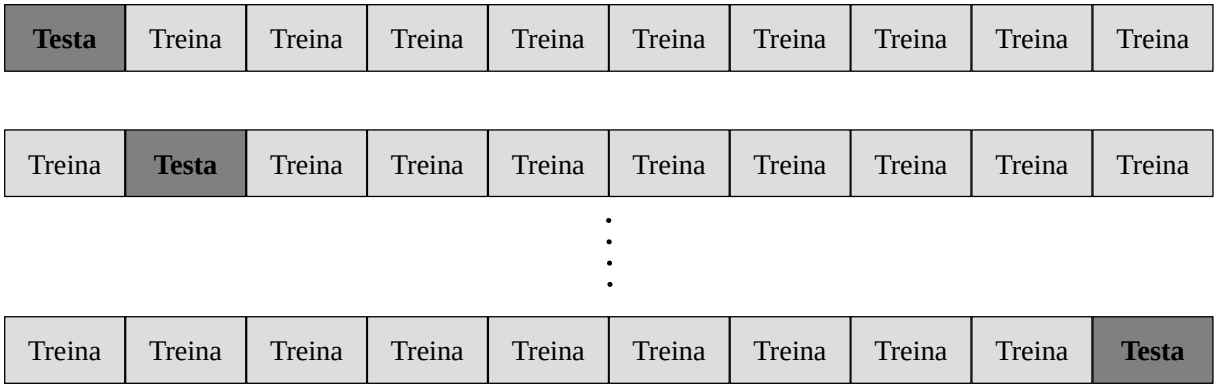


Figura 4.8: Ilustração da validação cruzada de 10-fold.

Um cuidado especial deve ser tomado aqui para evitar a contaminação dos dados de treinamento pelos dados de teste. Isso acontece quando a normalização de dados é realizada antes da validação cruzada, o que pode facilmente vazar informações sobre os dados de teste nos dados de treinamento, afetando a maneira como normaliza

o último. Com a contaminação, o modelo pode ter um desempenho melhor que o esperado quando comparado a situações em que essa informação não está disponível. O que deve ser feito é executar a normalização dentro da validação cruzada e somente nos dados de treinamento. Em sequência, as informações coletadas sobre os dados de treinamento podem ser usadas para a transformação dos dados de teste.

Para evitar essa contaminação, foi implementada uma função específica de normalização, como pode ser visto no Algoritmo 5.

Algoritmo 5: *Função de Normalização*

```
1 function [output, maxMin] = normalize_new(column,oldMaxMin)
2     if (length(oldMaxMin) == 0) then
3         maxMin = [max(column),min(column)];
4         output = (column - min(column)) / (max(column) - min(column));
5     else
6         output = (column - oldMaxMin(2)) / ( oldMaxMin(1) - oldMaxMin(2) );
7         maxMin = oldMaxMin;
8     end
9 end function
```

Como descrito acima, foi adotada uma validação cruzada de *k-fold* (k = 10), tomando nove dobras para treinar, onde a dobra remanescente foi usada para teste. Os dados foram normalizados usando a Equação 4.2, executando a normalização nas dobras de treinamento e usando os coeficientes obtidos para normalizar a dobra de teste. Desta forma, o máximo e o mínimo de cada coluna de cada treinamento foram extraídos e os dados de treinamento foram normalizados entre 0 e 1. Depois disso, a função retorna o máximo e o mínimo da coluna que foi identificada no treinamento. Esses valores são usados para normalizar as amostras de teste, usando a mesma equação.

Assim, as amostras de teste passam por uma normalização que foi aprendida exclusivamente no treinamento. Não há interferência das amostras de teste na normalização.

Além disso, as métricas de avaliação usadas para a comparação dos classificadores foram todas calculadas a partir da Matriz de Confusão (Sammut e Webb, 2011). Existem quatro valores mostrados em uma Matriz de Confusão e esses valores podem ser usados para calcular um número de métricas adicionais para a avaliação do classificador. Neste trabalho, as métricas são medidas da seguinte forma (Han et al., 2011):

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.3}$$

$$Precisão = \frac{VP}{VP + FP} \tag{4.4}$$

$$Sensibilidade = \frac{VP}{VP + FN} \tag{4.5}$$

onde os valores:

- Verdadeiros Positivos (VP): Estas se referem as tuplas positivas que foram corretamente rotuladas pelo classificador.
- Verdadeiros Negativos (VN): Estas são as tuplas negativas que foram corretamente rotuladas pelo classificador.
- Falsos Positivos (FP): Estas são as tuplas negativas que foram incorretamente rotuladas como positivas.
- Falsos Negativos (FN): Estas são as tuplas positivas que foram erroneamente rotuladas como negativas.

Acurácia representa os acertos do classificador como um todo, enquanto Precisão representa a taxa de acertos por classe para casos positivos. Ou seja, quantas instâncias positivas de uma determinada classe o classificador atingiu. Sensibilidade representa a taxa de uma amostra relevante sendo classificada corretamente.

A Taxa de Falso Positivo (TFP), que mostra a classificação de uma amostra negativa como positiva, e a Taxa de Falso Negativo (TFN), que mostra a classificação de uma amostra positiva como negativa, são medidas como segue.

$$TFP = \frac{FP}{FP + VN} \tag{4.6}$$

$$TFN = \frac{FN}{FN + VP} \tag{4.7}$$

Note que para a geração da Matriz de Confusão multiclasse, os classificadores foram avaliados considerando uma classe específica de cada vez, até as k classes identificadas: uma classe é fixa e admitida positiva e as classes k-1 são consideradas negativas. Este procedimento foi repetido para cada classe, com uma abordagem de um contra todos. Assim, uma Matriz de Confusão foi obtida para cada classe de forma que cada métrica geral do classificador fosse computada como a métrica média das k classes obtidas.

4.3.3.1 Resultados

Os experimentos foram realizados duas vezes: primeiro, com todas as 14 características (14f) do conjunto de dados; segundo, com todos as características redundantes removidas, considerando apenas 11 características (11f). Para os testes, os desempenhos do classificador Naive Bayes Gaussiano e do Algoritmo Proposto neste trabalho foram comparados com base em suas Matrizes de Confusão.

Todas as predições corretas para cada classe estão localizadas na diagonal principal de cada tabela do modelo (células verdes), e os erros de predição são representados pelos valores fora da diagonal principal (células vermelhas). Os outros valores são as métricas e taxas calculadas. Os valores mostrados são os resultados agregados de validação cruzada. Foi usado o método **classperf** do Matlab para extrair essa métrica dentro da validação cruzada, juntamente com o método **crossvalind** para obter os índices para dividir as dobras.

Para os experimentos com 14f, as Matrizes de Confusão do classificador Naive Bayes Gaussiano e do Algoritmo Proposto são mostradas na Tabela 4.7 e na Tabela 4.8, respectivamente.

Tabela 4.7: Matriz de Confusão para o classificador Bayesiano com todas as características **(14f)**.

	A Alvo	B Alvo	C Alvo	
A Saída	50,27 %	0,24 %	13,92 %	78,02 % 21,98 %
B Saída	0,22 %	21,19 %	0,27 %	97,75 % 2,25 %
C Saída	0,08 %	0,01 %	13,80 %	99,37 % 0,63 %
	99,41 % 0,59 %	98,85 % 1,15 %	49,31 % 50,69 %	85,26 % 14,74 %

A partir da Tabela 4.7, para o cenário utilizando todas as características, verifica-se que a precisão média produzida pela Naive Bayes Gaussiano atingiu uma classificação de **85,26%**, mas teve dificuldade de classificar as amostras da classe **C**. Ele apresentou uma alta taxa de falso negativo de **50,69%**, que é a taxa de amostras da classe **C** que não foram classificadas como classe **C**. Como resultado, o classificador não produziu uma classificação correta, uma vez que sua sensibilidade (**benefício**) foi baixa em **49,31%**.

Além disso, o classificador teve uma alta taxa de falsos positivos para a classe **A** de **21,98%**, o que significa que as amostras foram classificadas como streaming de vídeo, levando a uma Precisão de **78,02%** para a Classe **A**. No geral, para o problema de classificação proposto, este modelo apresentou um alto índice de classificação ruim de aproximadamente **14,74%** para as classes envolvidas.

Em contraste, o modelo do Algoritmo Proposto, Tabela 4.8, alcançou uma alta precisão de **98,79%** e não sofreu as mesmas dificuldades do modelo da Tabela 4.7.

Tabela 4.8: Matriz de Confusão do Modelo do Algoritmo Proposto com todas as características (**14f**).

	A Alvo	B Alvo	C Alvo	
A Saída	50,27 %	0,24 %	0,33 %	98,88 % 1,12 %
B Saída	0,23 %	21,19 %	0,33 %	97,44 % 2,56 %
C Saída	0,07 %	0,01 %	27,33 %	99,69 % 0,31 %
	99,40 % 0,60 %	98,83 % 1,17 %	97,65 % 2,35 %	98,79 % 1,21 %

O Algoritmo Proposto apresentou uma taxa baixa de elementos da classe **C** que não foram classificados como tal em aproximadamente **2,35%**, indicando uma alta sensibilidade ao classificar amostras dessa classe em **97,65%**. Além disso, houve uma redução significativa na taxa de falsos positivos para a classe **A** de **1,12%**. Consequentemente, este modelo obteve uma baixa taxa de erro para o classificador de **1,21%**.

Em outro cenário, sem características redundantes, o modelo Naive Bayes Gaussiano, apresentado na Tabela 4.9, obteve maior Acurácia em relação ao cenário anterior, apresentado na Tabela 4.7, atingindo **90,05%**. Entretanto, o modelo Naive Bayes Gaussiano teve maior dificuldade em classificar as amostras de Classe **C**, com alta taxa de falso negativo e sensibilidade muito baixa, que foram **33,72%** e **66,28%**, respectivamente.

Tabela 4.9: Matriz de Confusão para classificador Bayesiano sem características redundantes(**11f**).

	A Alvo	B Alvo	C Alvo	
A Saída	50,30 %	0,24 %	9,19 %	84,21 % 15,79 %
B Saída	0,20 %	21,19 %	0,25 %	97,93 % 2,07 %
C Saída	0,07 %	0,01 %	18,55 %	99,59 % 0,41 %
	99,47 % 0,53 %	98,84 % 1,16 %	66,28 % 33,72 %	90,05 % 9,95 %

Para o modelo do Algoritmo Proposto, apresentado na Tabela 4.10, obteve-se uma precisão ligeiramente melhor em comparação ao cenário anterior de **98,88%** e para todas as outras métricas comparadas ao cenário da Tabela 4.8. Como resultado, gerou-se uma baixa taxa de erro de **1,12%**.

Tabela 4.10: Matriz de Confusão do modelo do Algoritmo Proposto sem características redundantes (**11f**).

	A Alvo	B Alvo	C Alvo	
A Saída	50,30 %	0,24 %	0,35 %	98,82 % 1,18 %
B Saída	0,21 %	21,18 %	0,25 %	97,89 % 2,11 %
C Saída	0,06 %	0,01 %	27,39 %	99,75 % 0,25 %
	99,47 % 0,53 %	98,81 % 1,19 %	97,85 % 2,15 %	98,88 % 1,12 %

Considerando os resultados preliminares das métricas e as taxas geradas pela Matriz de Confusão, foi demonstrado que o modelo de Algoritmo Proposto obteve melhor desempenho e aplicabilidade média geral em relação aos resultados médios do outro modelo. No entanto, os resultados gerados podem ser comprometidos se o classificador tiver altos custos computacionais, por exemplo, na fase de classificação.

Neste trabalho, o custo computacional (ou tempo decorrido absoluto) é o tempo que o código leva para ser executado. O cálculo foi aplicado às fases de treinamento e teste para cada modelo e cenário avaliados.

Os experimentos foram realizados na versão Matlab R2016a (MATLAB, 2016) em um computador com processador Intel Core i7 com 16 GB de RAM DDR4.

A Figura 4.9 mostra os custos computacionais médios das fases de treinamento e classificação, respectivamente, para os modelos estudados.

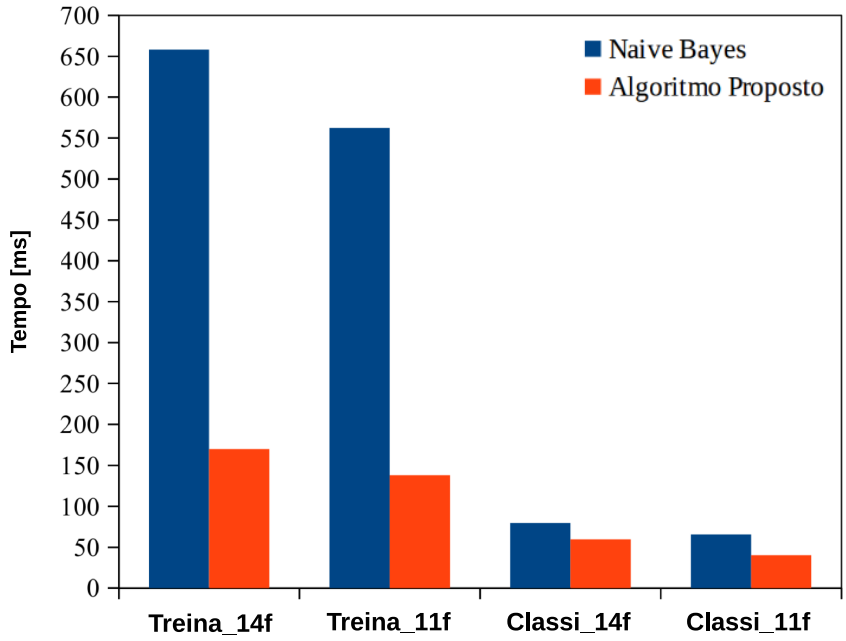


Figura 4.9: Custo Computacional (Treinamento = Treina e Classificação = Classi) dos Modelos com 14 e 11 características, respectivamente.

O modelo de Algoritmo Proposto apresentou menor tempo de treinamento em relação ao modelo Naive Bayes Gaussiano. O tempo de classificação é importante porque há casos em que o aumento do tempo de processamento dos pacotes diminui o desempenho do aplicativo em relação ao atraso fim-a-fim da rede.

Em (Baldi e Ofek, 2000), os autores apresentam um limiar de 100 ms para o atraso *end-to-end* das aplicações de voz. Assim, os classificadores de tráfego devem identificar um fluxo em um tempo relativamente menor que esse limite, pois o tempo total do atraso *end-to-end* corresponde à soma desse tempo de processamento com os valores de atraso de propagação e sobrecarga de roteamento, multiplicados pelo número de saltos entre o cliente e o servidor.

A Figura 4.9 também mostra os tempos de classificação para cada modelo e cenário que foram avaliados. O gráfico desta figura mostra que os dois modelos têm valores inferiores a 100 ms. No entanto, o modelo de Algoritmo Proposto apresenta valores

abaixo do modelo Naive Bayes Gaussiano para o cenário 14f (59 ms versus 79 ms) e para 11f (40 ms versus 65 ms) e muito menor que o valor 100 ms estabelecido por aplicações não elásticas , como streaming de vídeo (Baldi e Ofek, 2000).

A análise dos algoritmos deve levar em conta tanto o custo total (tempo total decorrido para as fases de treinamento e classificação) quanto o custo médio de classificação (valor médio de todas as interações computadas da fase de classificação do experimento). Quanto mais baixos forem esses valores, melhor será o modelo, pois será capaz de classificar o tráfego de vídeo em tempo real. O inverso indica o pior classificador para o cenário. Dessa forma, o modelo Naive Bayes Gaussiano obteve os maiores custos computacionais para a fase de classificação. Por outro lado, o Algoritmo Proposto gerou um pequeno impacto no atraso total, com menos efeito sobre os aplicativos que estão sujeitos a restrições de tempo, como o vídeo.

O objetivo desses experimentos foi avaliar a viabilidade de um classificador de tráfego de rede usando informações de cabeçalho de pacote. O algoritmo Naive Bayes Gaussiano, apesar de atingir **90%** de Acurácia com as características selecionadas, obteve a pior taxa de Precisão de apenas **66,28%** das amostras da Classe **C**, que também são importantes para o balanceamento de carga.

Os resultados do modelo classificador com o Algoritmo Proposto atingiram uma Acurácia muito alta, o que sugere uma superioridade da aplicação neste cenário, mas estudos adicionais ainda precisam ser realizados com outros métodos encontrados na literatura (Kaoprakhon e Visoottiviseth, 2009; Ubik e Žejdl, 2010; Gomes e Madeira, 2012; Singh et al., 2013; Dong et al., 2013; Nascimento et al., 2014; Yamansavascilar et al., 2017; Sun et al., 2018b) para confirmar essa tendência.

Com relação ao tempo, verificou-se que um modelo com menos características levou aproximadamente 75% menos tempo do que o necessário para o treinamento em comparação a um modelo com todas as características, embora esses tempos fossem baixos. Isso ocorreu para os dois modelos de classificadores.

Em relação à Acurácia, o algoritmo do modelo Naive Bayes Gaussiano apresentou uma melhora significativa do desempenho ao reduzir o número de características. Muito provavelmente, ao considerar a hipótese ingênua de que as variáveis de entrada são independentes, sua precisão foi bastante afetada pelas variáveis fortemente correlacionadas. O mesmo não foi observado no modelo Algorítmico Proposto, cujo desempenho foi semelhante para ambos os cenários, uma vez que já considera em sua implementação que seus atributos de entrada não são independentes.

4.4 Considerações Finais

Neste capítulo, foi apresentado o detalhamento para o desenvolvimento e a avaliação de um módulo de classificação de tráfego de streaming de vídeo, como uma solução inteligente para cenários que precisam de tratamento de tráfego adequado, conforme exigido por aplicativos específicos. Os resultados mostraram que a abordagem foi promissora e inovadora em diferentes aspectos.

O módulo de classificação adotou uma nova abordagem para o relaxamento da hipótese de independência entre atributos do algoritmo Naive Bayes. Em relação à classificação de pacotes do tráfego de streaming de vídeo, a Acurácia média para este Algoritmo Proposto, **98,88%**, mostrou que o módulo proposto é promissor para ser aplicado em cenários em tempo real. Além disso, produziu um melhor desempenho com baixos custos computacionais, tornando-se promissor como modelo de classificação multiclasse.

O melhor classificador do trabalho (Yamansavascilar et al., 2017) obteve um desempenho médio global de **93,94%** de Acurácia para os aplicativos de tráfego listados. No entanto, para aplicativos de streaming de vídeo, como o Vimeo e o YouTube, o melhor classificador obteve precisões de **85,20%** e **90,40%**, respectivamente. O trabalho (Gomes e Madeira, 2012), que adotou o Naive Bayes Gaussiano como classificador, atingiu um nível de Acurácia de **95%**. No trabalho (Kaoprakhon e Visoottiviseth, 2009), que classificou por categoria e não por tipo de aplicação, o classificador obteve uma Acurácia média geral de **98%**, mas uma baixa sensibilidade ao classificar amostras da categoria de vídeo de **78%**. Os resultados apresentados em (Sun et al., 2018b) mostraram que o TrAdaBoost, que é baseado no modelo de aprendizagem por transferência, apresenta um desempenho melhor do que os métodos comparados em termos de precisão (**98,7%**). No entanto, eles usaram um conjunto de dados de 2005 contendo apenas tráfego de dados tradicional sem multimídia. As técnicas de transferência de aprendizado foram aplicadas principalmente a aplicações de pequena escala com uma variedade limitada, como localização baseada em redes de sensores, classificação de texto e problemas de classificação de imagem. Várias questões importantes de pesquisa ainda precisam ser abordadas para satisfazer outras aplicações desafiadoras, como classificação de vídeo, análise de redes sociais e inferência lógica (Pan e Yang, 2010).

Os trabalhos (Yamansavascilar et al., 2017; Kaoprakhon e Visoottiviseth, 2009; Sun et al., 2018b) não apresentaram os custos computacionais dos classificadores utilizados. O trabalho (Gomes e Madeira, 2012) obteve uma classificação de tráfego em 5 ms para um conjunto de dados de 4.336 pacotes coletados. Nosso Algoritmo Proposto alcançou um tempo de classificação de 40 ms para um conjunto de dados muito maior de

551.095 pacotes. Esse valor está bem abaixo do limite aceitável para tráfego em tempo real de 100 ms (Baldi e Ofek, 2000). Alternativas podem ser mais caras. Support Vector Machines (SVMs), por exemplo, é uma abordagem com altos custos computacionais para um grande conjunto de dados, uma vez que foi originalmente projetado para classificação binária. Como o problema neste trabalho envolve a classificação multiclasse, seria necessário usar mais de um classificador SVM, como "um contra todos", o que aumentaria ainda mais a sobrecarga (Hsu e Lin, 2002).

Outro aspecto importante é que o presente trabalho aplicou o aprendizado de máquina para classificar o tráfego em tempo real e evitou o uso de portas TCP/UDP do cabeçalho do pacote, o que não funciona nos cenários atuais com a transmissão via protocolo HTTP. A classificação foi capaz de distinguir entre streaming de vídeo da Netflix, streaming de vídeo do YouTube e tráfego em segundo plano.

Em relação à geração de dataset, os scripts de captura eram compatíveis com o agendador **CRON**. Portanto, eles podem ser usados em futuras implementações do módulo classificador para capturar novos tráfegos e treinar um novo modelo de classificação a cada hora, por exemplo. Além disso, a análise dos dados coletados revelou que algumas variáveis do pacote IP são altamente correlacionadas, o que pode interferir no modelo de classificação, dependendo do classificador escolhido. Nesse sentido, a metodologia adotada para seleção de atributos mostrou-se adequada e vantajosa, gerando um bom *trade-off* entre os aspectos de desempenho e custo computacional.

Capítulo 5

Validação do Módulo de Classificação de Tráfego em uma Arquitetura SDN

5.1 Introdução

Este capítulo descreve a integração do módulo de classificação de tráfego de streaming de vídeo, apresentado no Capítulo 4, à uma arquitetura SDN em um cenário de rede Data Center. Também são apresentados os casos de testes executados e os resultados obtidos, para avaliar a arquitetura proposta.

5.2 Módulo de Classificação de Tráfego On-line

No Capítulo 4, foi apresentada a abordagem metodológica adotada para o módulo de classificação de tráfego de streaming de vídeo baseado em aprendizado de máquina. Como descrito, a arquitetura inicial proposta, bem como o desenvolvimento dos algoritmos, funções de treinamento e teste, implementados no algoritmo de classificação proposto (AP) (Seção 4.2.2), foram codificados usando a linguagem de programação M-code da ferramenta Matlab R2016a (MATLAB, 2016).

Contudo, para utilizar o módulo de classificação proposto em uma arquitetura de Redes Definidas por Software, foi necessário codificá-lo em uma linguagem compatível com o Controlador SDN utilizado na arquitetura proposta. Sendo assim, o módulo de classificação foi re-codificado utilizando as bibliotecas da linguagem Python, o que permitiu acoplá-lo como um módulo do controlador SDN com o protocolo OpenFlow da arquitetura proposta.

Com a nova implementação, o módulo passou a ser denominado de Módulo de Classificação de Tráfego On-line, com seu blocos e scripts específicos descritos a seguir. A Figura 5.1 apresenta a visão geral do funcionamento do Módulo de Classificação de Tráfego On-line.

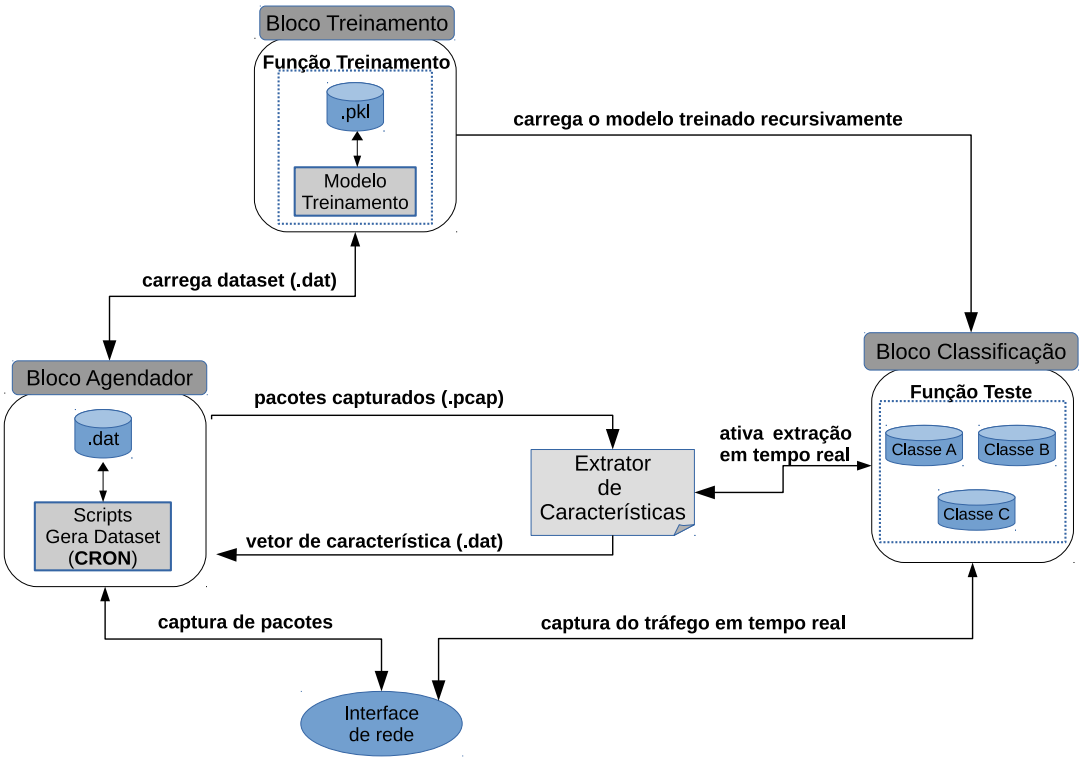


Figura 5.1: Funcionamento do Módulo Classificador de Tráfego On-line.

5.2.1 Bloco Agendador

Esse bloco contém scripts desenvolvidos para capturar e gerar o conjunto de dados, considerando cada tipo de aplicação, para auxiliar em tempo real o bloco de treinamento. Três scripts shell bash foram desenvolvidos (considerando cada tipo de serviço do tráfego: Netflix, YouTube e Background) e implementados para capturar e gerar a base de dados para o modelo de classificação. O Bloco Agendador foi configurado para produzir uma captura de 2 minutos de tráfego. Os dados extraídos pelo extrator de características são concatenados e salvos em dois arquivos, uma matriz de características e um vetor de rótulos. Ressalta-se que o desenvolvimento do bloco levou em conta sua execução pelo **CRON**, uma ferramenta do Sistema Operacional UNIX que permite o agendamento de tarefas.

5.2.2 Bloco Treinamento

Esse bloco foi desenvolvido para treinar o Algoritmo Proposto, a ser utilizado no módulo de classificação. A etapa do treinamento é fundamental não apenas para preparar o modelo de classificação, mas para aprimorar periodicamente seu desempenho de predição. Contudo, é necessário sempre fazer mudanças na maneira como treinamos o modelo, para melhor otimizá-lo. Como é necessário periodicamente testar o modelo, é preciso uma maneira de salvar o modelo treinado no controlador SDN. Fazendo isso, estaremos armazenando os coeficientes e algumas informações relacionadas do classificador modelado (treinado).

Para a funcionalidade do bloco de treinamento, foi implementado na função de treinamento do Algoritmo Proposto o módulo **Joblib**[1] do Python. Esse módulo possui um conjunto de ferramentas que fornece **Pipelining** leve e cache de disco transparente dos valores de saída. O armazenamento em cache transparente é rápido. Na estrutura do Bloco de Treinamento, o Joblib possibilita treinar um modelo do Algoritmo Proposto no conjunto de dados gerado pelo Bloco Agendador, salvar o modelo em um arquivo (**.pkl**[2]) e carregá-lo, apenas se necessário, para fazer previsões no conjunto de dados de testes não vistos (tráfego em tempo real).

5.2.3 Bloco Classificação

Esse outro bloco foi desenvolvido para utilizar o Algoritmo Proposto (4.2.2), operando no tráfego ativo da rede. O algoritmo foi aqui implementado de forma a não extrair características de endereços IP, número de porta ou cabeçalhos HTPP, TCP ou UDP, ou cargas úteis (payload) do tráfego de rede. Portando, o sistema se torna robusto para: encriptação; web proxies; mudanças no número de portas; mascaramento de IP; e fragmentação de pacotes.

Uma funcionalidade importante implementada no bloco é a possibilidade dele carregar o modelo do classificador treinado periodicamente a partir do arquivo **.pkl**, construído no bloco treinamento, antes de iniciar a classificação em tempo real. Com isso, é possível acompanhar possíveis mudanças nas características do tráfego em tempo real, treinando com um novo conjunto de dados periodicamente. O bloco de Classificação é transparente para as camadas de transporte e aplicação, e portanto, se encaixa nos requisitos de um classificador de tráfego de streaming de vídeo.

Esses blocos operam da seguinte maneira. Os pacotes capturados pelos shell scripts de cada serviço são salvos em arquivos contendo um identificador com o nome do video

[1] https://pypi.org/project/joblib/
[2] https://docs.python.org/3.4/library/pickle.html

ou tráfego de background e a data do inicio da captura, no formato **.pcap**. Ressalta-se que o bloco foi agendado para que os seus scripts sejam executados periodicamente, a cada 24h, possibilitando gerar um novo conjunto de dados e um novo modelo de treinamento.

Os pacotes capturados são concatenados em um único arquivo e passados para o extrator de características, o qual retorna os vetores de características e rótulos correspondentes (**.dat**). Estes vetores são então passados para função de treinamento, o qual tem a responsabilidade de gerar um modelo treinado (**.pkl**) para o classificador. Também foi implementada uma rotina no bloco de treinamento que, após a geração de um novo conjunto de dados pelo Bloco Agendador, ele seja reiniciado para atualizar com o novo modelo do classificador treinado. Ao executar o Bloco de Classificação, o modelo de classificação treinado é carregado para validar a função de teste implementada para classificar o tráfego capturado em tempo real. A funcionalidade de extração de características (script) foi implementada no bloco e combinada com os módulos Scikit-Learn[3] e Scapy[4] (pcapy) do Python, para capturar, extrair características e classificar o tráfego em tempo real.

Com o proposito de facilitar a criação de um módulo para o controlador SDN, foi desenvolvido um sistema em tempo real, integrando os três blocos descritos acima, para classificar o tráfego efetivamente. A interface do sistema é mostrada na Figura 5.2.

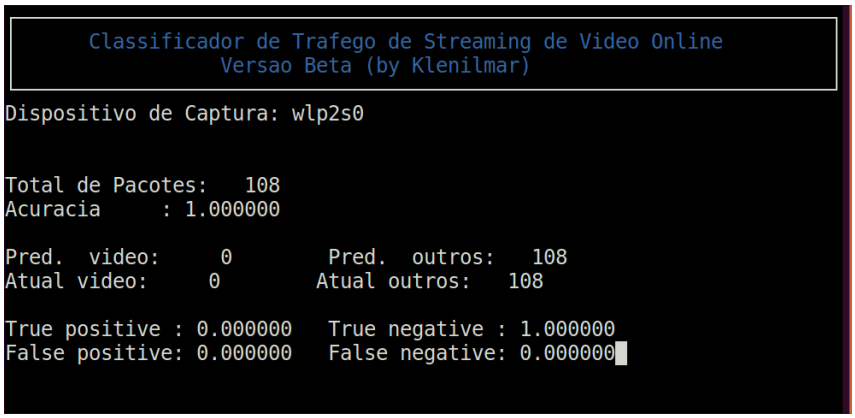


Figura 5.2: Classificador de tráfego em tempo real em operação.

[3]https://scikit-learn.org/stable/
[4]https://pypi.org/project/scapy-python3/

5.3 Arquitetura do Controlador SDN com Classificador On-line

A arquitetura do Controlador SDN com Classificador On-line é apresentada na Figura 5.3. O Módulo de Classificação de Tráfego On-line (Seção 5.2) foi integrado a uma arquitetura SDN com protocolo OpenFlow. A arquitetura funciona da seguinte forma: periodicamente, o Monitor de Rede solicita ao controlador uma captura de tráfego. O intervalo de tempo para esta solicitação é configurado por demanda. Em seguida o controlador reúne as informações de tráfego e responde ao Monitor de Rede. Esta informação é então encaminhada para o Bloco Agendador e, na sequência, são acionadas as funcionalidades dos demais blocos. O extrator de características é implementado juntamente com o bloco de classificação, assemelhando-se a um *wrapper*. O classificador constrói o modelo de classificação e processa os fluxos, conforme descrito na Seção 5.2. Quando os resultados da classificação forem gerados, o classificador envia as tabelas de fluxos para os switches OpenFlow da rede SDN.

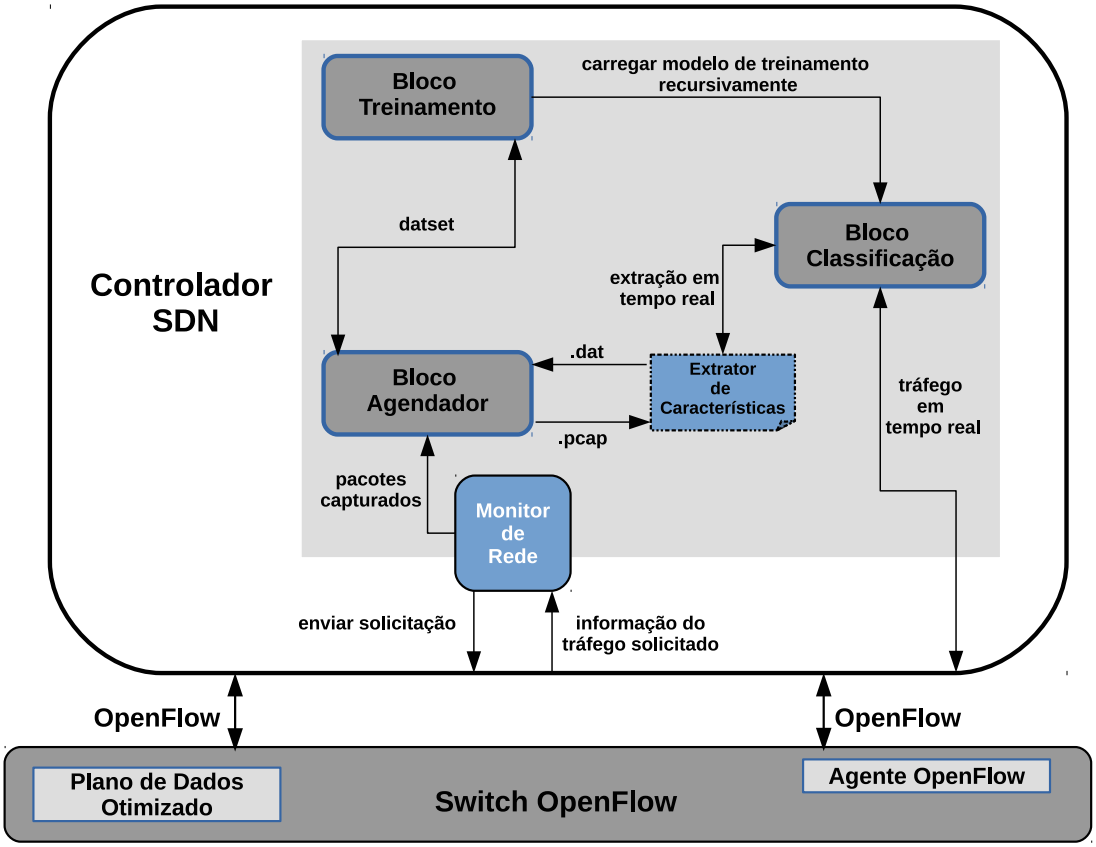


Figura 5.3: Arquitetura do Classificador de Tráfego SDN.

Importante ressaltar que esta é um proposta de uma arquitetura modular, na qual qualquer dos blocos descritos pode ser alterado livremente sem que sejam necessárias grandes alterações nos demais blocos. Outra característica importante no desenvolvimento desta arquitetura é a possibilidade do classificador poder suportar a implementação de uma variedade de algoritmos de classificação baseados em aprendizado de máquina.

5.3.1 Estendendo o Contador de Tráfego OpenFlow

A SDN oferece uma reformulação da lógica do controle de rede e atenua as limitações impostas pelas redes IP tradicionais. Na abordagem SDN, todas as decisões de controle são tomadas pelo controlador, enquanto os dispositivos de rede tornam-se encaminhadores de pacotes, programáveis por meio do protocolo OpenFlow (Wickboldt et al., 2015). No entanto, apesar dos benefícios trazidos pela SDN, ainda há uma falta de suporte adequado para realizar tarefas relacionadas à classificação de tráfego em redes baseadas no OpenFlow (Xie et al., 2018). Alcançar alta precisão na classificação de tráfego na SDN é difícil por alguns motivos: (i) existem perfis de tráfego muito semelhantes, o que dificulta e aumenta o custo de sua classificação, por exemplo, os fluxos HTTP e DNS são caracterizados por rajadas de pacotes; (ii) há uma falta de suporte para determinar qual é o conjunto ideal de características de fluxo para caracterizar diferentes tipos de perfis de tráfego; e (iii) as características de fluxos nativos disponíveis no OpenFlow, como contagem de pacotes e bytes, não transmitem informações suficientes para distinguir com precisão entre alguns tipos de fluxos, pois esses contadores não permitem detectar comportamento de tráfego atípico, como rajadas de pacotes.

Considerando as questões acima mencionadas, neste trabalho foi desenvolvida uma extensão dos contadores nativos do OpenFlow, aproveitando as características estatísticas obtidas pelo componente Extrator de Características do módulo, de forma a proporcionar informações mais precisas sobre o comportamento dos fluxos de tráfego ao longo do tempo. O Algoritmo 5 descreve a implementação da extensão proposta.

A estratégia implementada, iniciada pelo componente Monitor de Rede, é usar proativamente o OpenFlow para instalar um evento **Flow_Entry** (entrada de fluxo) que instrui o switch OpenFlow a encaminhar o tráfego para o controlador SDN. É definido um número (N) inicial curto de pacotes do tráfego a ser encaminhado para o controlador. Quando N pacotes chegam ao controlador, através de uma mensagem OpenFlow do evento **Packet_In** (pacote de entrada), o bloco de classificação entra em ação aplicando uma função denominada de processamento de pacotes do extrator de

características, com dois métodos: um para identificar características padrões de cada pacote e outro para identificar características extraídas de uma janela de pacotes. O uso de um evento **Flow_Entry** é para reduzir o número de mensagens do **Packet_In** no controlador.

Algoritmo 6: EXTENSAO_OPENFLOW

Entrada: N Pacotes

Saída: Fluxo Identificado

1 **início**

2 **Passo 1:** Instalar evento no switch OpenFlow (**Flow_Entry**)

3 **Passo 2:** Encaminhar N Pacotes ao Controlador (**Packet_In**)

4 **Passo 3:** Ativar Função Processamento de Pacotes

Metodo 1: Indentifica características de cada pacote

Metodo 2: Indentifica características de um janela

Passo 4: Atualizar tabela de fluxos (**Flow_Mod**)

5 **fim**

Usando informações reunidas após o bloco de classificação ter atuado, o controlador age imediatamente nesse fluxo aplicando as regras de encaminhamento. A tabela de fluxos do switch OpenFlow é organizada para enviar o tráfego pelos N servidores réplicas, em vez de um único servidor com base na relação: categoria do tráfego (classe) -> servidor réplica específico. Isso é realizado enviando uma mensagem **FlowMod** ao switch OpenFlow, atualizando as informações da tabela de fluxo para balancear os pacotes de entrada para o destino definido na classificação de tráfego.

Por padrão, o switch OpenFlow não encaminha o pacote IP inteiro junto com a mensagem de controle. Para tal, foi preciso ativar um módulo nativo do controlador SDN (*misc.full_payload*), para ser possível o encaminhamento completo dos pacotes IP para o controlador SDN.

5.4 Ambiente Experimental e Resultados

Esta seção descreve o ambiente em que a arquitetura foi avaliada, o problema abordado e os parâmetros adotados.

5.4.1 Ambiente de Execução

Data Centers são estruturas onde vários servidores e equipamentos de comunicação são co-localizados, devido a seus requisitos ambientais e de segurança física em comum, bem como facilidade de manutenção (Barroso et al., 2013).

Atualmente, a arquitetura típica de Data Center consiste em uma árvore de dois ou três níveis de switches ou roteadores. Com o objetivo de facilitar o entendimento do ambiente, neste trabalho foi adotada uma arquitetura denominada *Two Tier* (duas camadas) (Al-Fares et al., 2008; Barroso et al., 2013). Na arquitetura de duas camadas, o Data Center é representado apenas por roteadores, switches núcleo compondo a primeira camada, switches de borda formando a segunda camada, servidores e enlaces entres esses elementos, conformes ilustrado pela Figura 5.4.

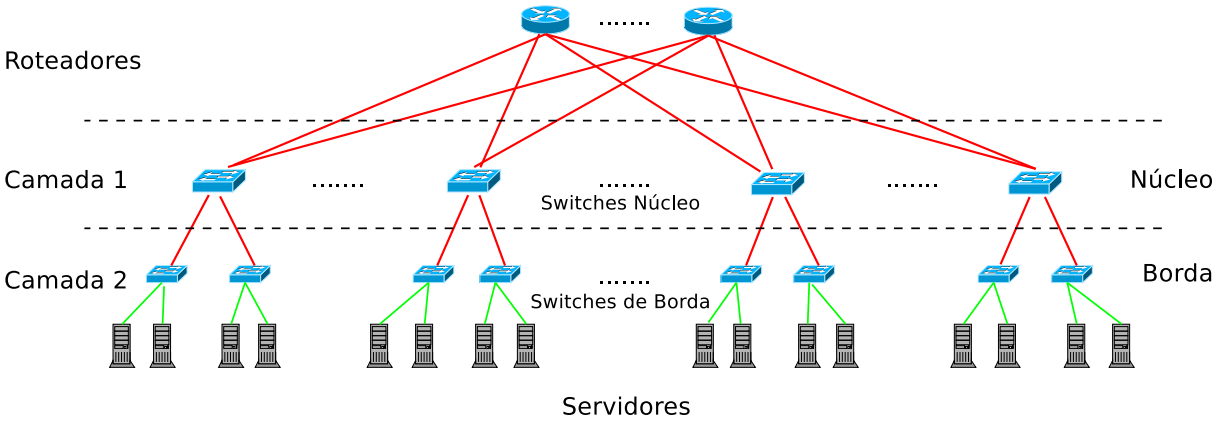


Figura 5.4: Arquitetura típica de Data Center de duas camadas (adaptada de (Al-Fares et al., 2008)).

Na arquitetura de duas camadas os roteadores, interfaces de conexão entre o ambiente do Data Center e a Internet, estão conectados a vários switches núcleo. Cada switch núcleo se conecta a vários switches de borda. Cada switch de borda se conecta a vários servidores (denominados de réplicas) (Barroso et al., 2013), sendo que cada servidor se conecta a um único switch de borda. O tráfego externo ao Data Center com os clientes na Internet, proveniente dos roteadores, é denominado tráfego de entrada. O tráfego entre os servidores é chamado de tráfego interno. Essas duas camadas possibilitam que os servidores comuniquem-se entre si e com a Internet, através de suas interfaces de rede.

A região de interesse neste trabalho é constituída por um switch de borda e os servidores réplica a ele ligados. Assim, definimos como elemento principal o switch

de borda, que submetido ao tráfego de entrada fará o balanceamento de carga para os servidores réplicas, seguindo o seguinte fluxo: tráfego de entrada -> switch -> servidores réplica. A Figura 5.5 apresenta a arquitetura aqui proposta.

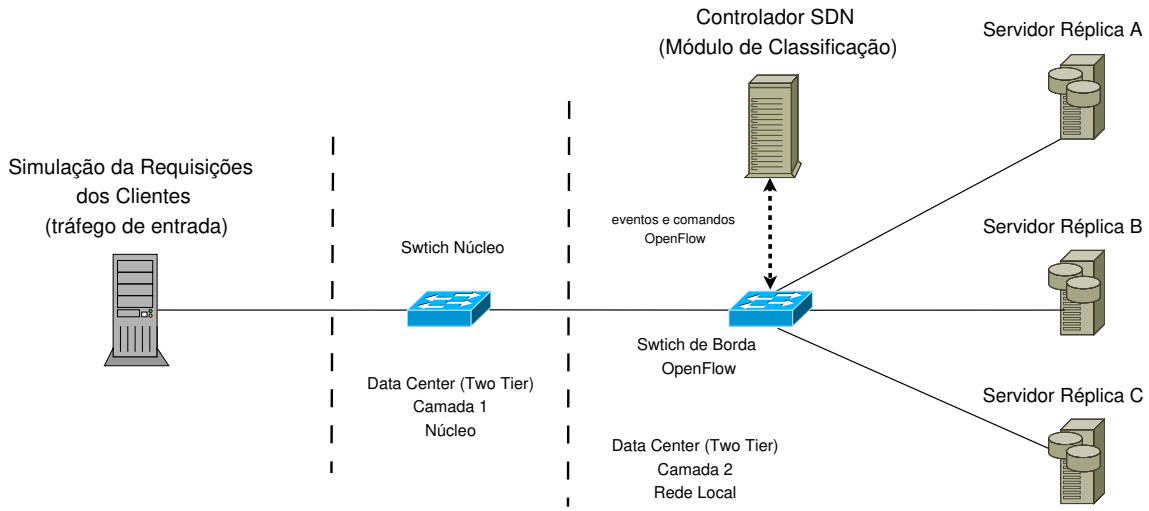


Figura 5.5: Arquitetura do Data Center com N = 3 (3 servidores e 3 enlaces).

Para a realização dos experimentos foi montado um ambiente consistindo em: uma rede organizada com topologia de uma rede local de um Data Center virtual, criada a partir do emulador de redes Mininet[5], na camada de encaminhamento, com o controlador POX[6] versão 0.2.0 (Carp), no plano de controle.

Mininet é uma ferramenta de emulação, capaz de criar topologias de rede completas com hosts, enlaces e switches em uma única máquina, de maneira customizada utilizando a linguagem Python. Através da interface CLI (*Command Line Interface*) do Mininet é possível interagir facilmente com a rede SDN, personalizá-la, compartilhá-la, ou simplesmente implantá-la em um hardware real. O software emulador de switch utilizado dentro do Mininet é o Open vSwitch[7]. Ele foi criado para poder ser utilizado como um switch comum, com protocolo OpenFlow habilitado, dentro de ambientes virtuais.

O controlador POX foi desenvolvido baseado no modelo do controlador NOX[8], totalmente na linguagem Python. POX é adequado aos casos em que a escalabilidade da rede não seja um requisito crítico. Facilita a prototipação rápida e desenvolvimento de aplicações de teste para OpenFlow. Ele é constituído por inúmeros módulos, o que

[5]http://mininet.org/
[6]http://www.noxrepo.org/pox
[7]https://www.openvswitch.org/
[8]https://github.com/noxrepo/nox

facilita a criação e adição de novos módulos, como foi realizado com o módulo de classificação de tráfego on-line.

O ambiente foi preparado utilizando um computador com processador Intel Core i7 com 16GB de RAM com Sistema Operacional Ubuntu 16-04 LTS. No mesmo computador foram instalados o controlador POX e o Mininet, por questões de desempenho.

5.4.2 Cenário 1 - Desempenho do Classificador SDN

Este cenário teve como objetivo testar o desempenho do módulo de classificação de tráfego na arquitetura de rede SDN com tráfego ativo, objetivando o monitoramento, tratamento do tráfego e a programação na rede do Data Center.

Neste cenário, foi explorada a viabilidade da classificação antecipada do tráfego capturando os N primeiros pacotes consecutivos, variando N, conforme orienta a pesquisa em (Peng et al., 2015). Para medir o desempenho da classificação, usamos duas métricas que são amplamente usadas para avaliação de desempenho em classificação de tráfego (Kim et al., 2008): acurácia geral do Classificador (Seção 4.3.3); e *F-measure*, que é uma média harmônica entre precisão e sensibilidade (Seção 4.3.3), de modo a trazer um número único que indique a qualidade geral do seu modelo e que trabalhe bem até com conjuntos de dados que possuem classes desproporcionais. Essa última métrica foi usada para avaliar o desempenho por classe. A *F-measure* é calculada por:

$$F - measure = \frac{2 \cdot precisão \cdot sensibilidade}{precisão + sensibilidade} \tag{5.1}$$

Neste cenário, foram utilizados e configurados no módulo de classificação SDN dois outros algoritmos de aprendizado de máquina, para avaliação e comparação com o Algoritmo Proposto (AP): Naive Bayes Gaussiano (NBG) e Support Vector Machine (SVM). Os tráfegos de fluxos gerados na rede SDN foram de três tipos, todos com duração de 2 minutos: (i) streaming de vídeo da Netflix; (ii) streaming de vídeo do YouTube e (iii) tráfego de background (downloads de arquivos, Web Browser, e-mail).

As Figuras 5.6 e 5.7 apresentam a acurácia média geral dos três algoritmos de aprendizado de máquina ao utilizar os primeiros N pacotes consecutivos, variando N de um a dez, e aplicados com os dois grupos de características selecionados pelo método de seleção proposto neste trabalho (Seção 4.3.2): com 14 características e com 11 características.

Os resultados mostram que o Algoritmo Proposto (AP) e o NBG atingiram as maiores acurácias em relação ao SVM. Em particular, para 14 características, eles atingiram uma acurácia média geral de 98%, 84,12% e 63,19%, respectivamente.

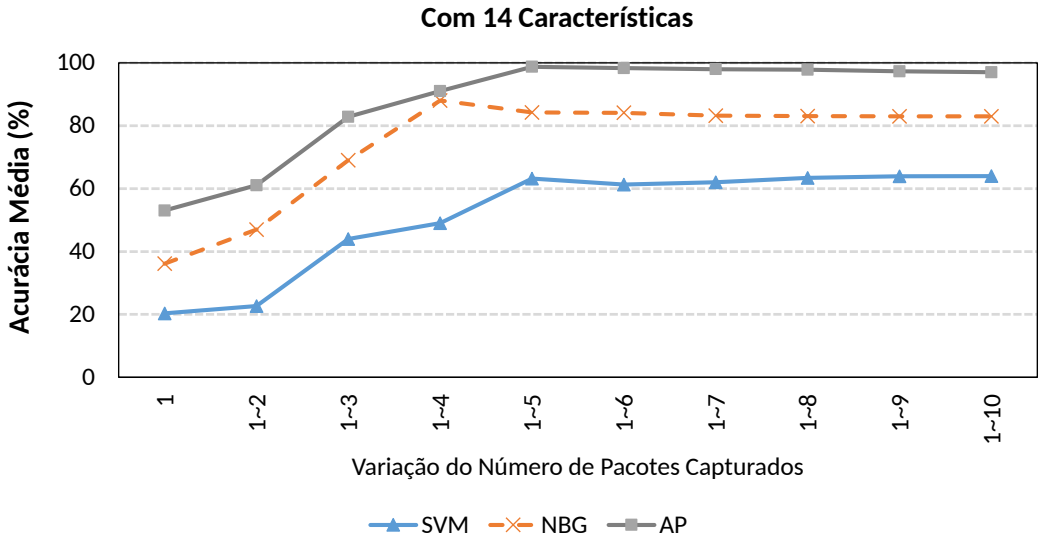


Figura 5.6: Acurácia geral média dos algoritmos com 14 características para os N primeiros pacotes (N variando de 1 a 10).

Com 11 características, o modelo AP manteve-se com uma ótima acurácia média, 98,6%, mas os modelos NBG e SVM sofreram um aumento considerável em suas acurácias, passando de 84,12% para 89,88% e de 63,19% para 71,12%, respectivamente. Mesmo com o aumento da acurácia médias dos dois outros modelos, o modelo com AP, manteve a melhor precisão para a classificação.

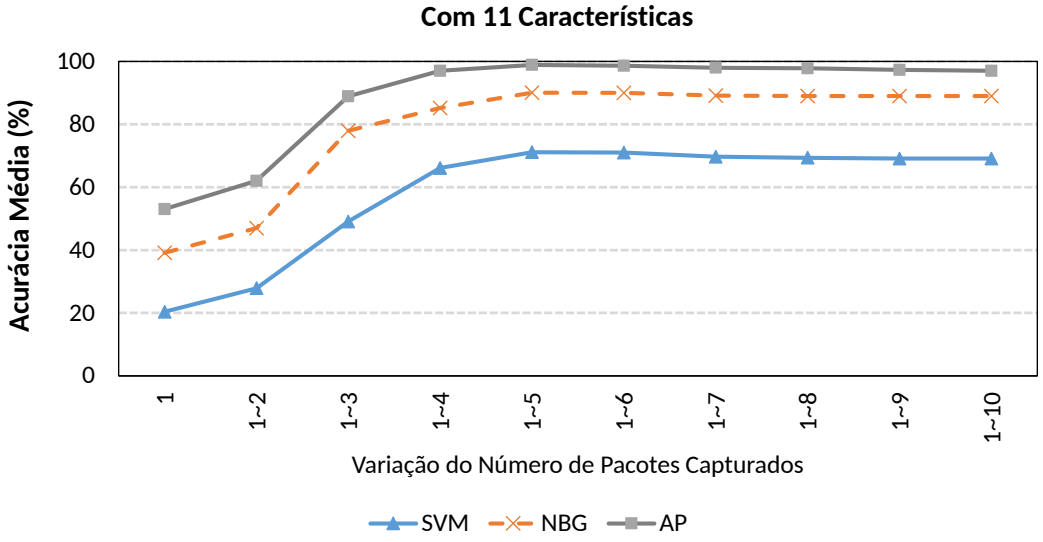


Figura 5.7: Acurácia geral média dos algoritmos com 11 características para os N primeiros pacotes (N variando de 1 a 10).

Este experimento revelou um comportamento relevante: que a acurácia atinge seu máximo para determinada faixa de N. A Tabela 5.1 apresenta as acurácias médias dos primeiros N pacotes consecutivos, N variando de 1~10, para o cenário com melhores resultados (11 características).

Tabela 5.1: Acurácia dos algoritmos com 11 características.

	ACURÁCIA (%)		
Número de Pacotes	**SVM**	**NBG**	**AP**
1	20,39	39,13	53,05
1~2	23,73	46,99	62,00
1~3	49,01	77,92	83,00
1~4	66,12	85,21	91,89
1~5	71,12	90,05	98,88
1~6	71,00	90,00	98,60
1~7	69,70	89,12	98,40
1~8	69,37	89,05	98,00
1~9	69,10	89,00	97,80
1~10	69,02	89,00	97,30

Os modelos atingiram as maiores acurácias quando usados com os tamanhos dos primeiros cinco ou seis pacotes. A adição do sétimo ao décimo tamanho de pacote não aumenta o desempenho de classificação dos algoritmos, chegando a reduzi-lo ligeiramente.

Foi possível constatar que as informações estatísticas de certa quantidade de pacotes, na maioria das vezes, geram maior acurácia do que pacotes individuais. Os tamanhos dos dois primeiros pacotes contribuem mais na classificação de aplicações com tráfego UDP, enquanto a partir do segundo ao sexto pacote, contribuem mais para classificação de aplicações com tráfego TCP. Isso ocorre porque os aplicativos com TCP concluem sua fase de negociação de conexão no primeiro ou no segundo pacote em um fluxo unidirecional, portanto, o segundo ou terceiro pacote a seguir normalmente tem um tamanho de pacote específico do aplicativo.

Em particular, a classificação com o tamanho do primeiro ao quarto pacote causa uma queda abrupta na acurácia média geral, para todos os modelos de classificadores. Neste sentido, ficou comprovado que os primeiros seis pacotes consecutivos são os mais relevantes e desempenham um papel importante na classificação precisa do tráfego para as classes com aplicações de streaming de vídeo (Netflix e YouTube). No caso de fluxos de e-mail e Web (*tráfego de background*), o primeiro tamanho de pacote não

contém nenhuma informação útil para classificação precisa, pois todos são gerados na fase de configuração de conexão TCP pré-definida.

A partir dos resultados obtidos acima, seguiu-se para analisar o custo computacional, aplicando a classificação nos primeiros seis pacotes consecutivos. A Figura 5.8 mostra o tempo de classificação para os três algoritmos, com N = 6 primeiros pacotes.

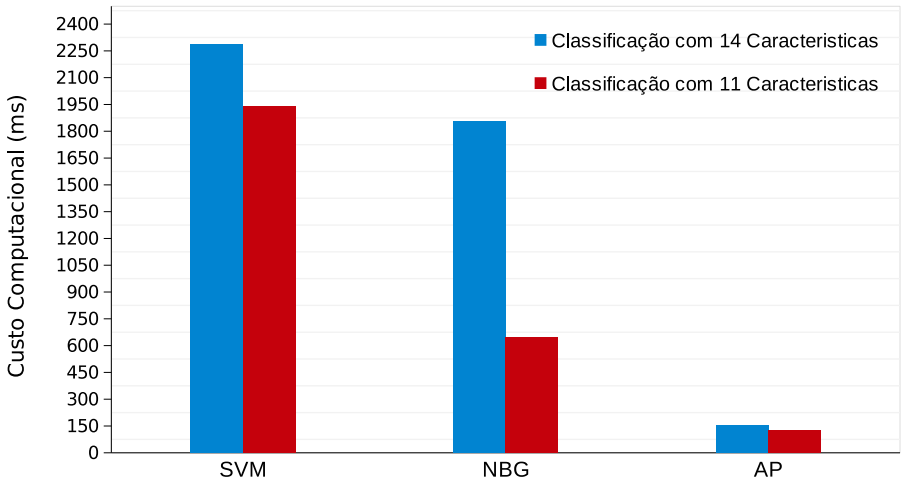


Figura 5.8: Custo Computacional para classificação.

É claramente difícil definir de maneira justa o melhor algoritmo, com base apenas em sua acurácia, precisão, sensibilidade e na sua *F-measure*. O custo computacional é particularmente importante, quando se considera a classificação em tempo real de potencialmente milhares de fluxos simultâneos na rede.

O Algoritmo Proposto (AP) foi o que obteve o menor custo computacional de classificação (154 ms e 125 ms), seguido do NBG (1.853 ms e 645 ms) e do SVM (2.289 ms e 1.939 ms), para cenários com 14 e 11 características, respectivamente. Entretanto, em (Baldi e Ofek, 2000), foi identificado que o limiar estabelecido para que os classificadores possam atuar em tempo real, em fluxos de tráfego carregando aplicações não elásticas, é de 100 ms.

Todos os modelos ultrapassaram esse limiar. Porém, é preciso ressaltar que por estar sendo utilizado software de emulação, o switch SDN é implementado em software (Open vSwitch), apresentando baixo desempenho de processamento em relação ao um switch implementado em hardware. Considerando o comportamento observado do switch SDN emulado, é esperado melhor desempenho em switches SDN físicos. Com esta análise, o algoritmo AP obteve o menor custo para classificação com proximidade para um limiar aceitável no mundo real. Por fim, algoritmos de ML que foram modelados para resolver problemas de classificação binária geram um tempo de classificação

muito grande para uma base de dados extensa, quando aplicados para problemas multiclasse (como o tratado neste trabalho). O algoritmo SVM, por exemplo, gerou um tempo de classificação muito longe do limiar aceitável para problemas de classificação de streaming. Como no problema tratado existem três classes, foi preciso adotar abordagem que envolva o uso de mais de um classificador SVM, um contra todos, por exemplo, que aumentou ainda mais o overhead.

Finalizando os testes de desempenho do módulo de classificação de tráfego na arquitetura SDN, foram analisados os resultados da métrica *F-measure*. As Figuras 5.9 e 5.10 apresentam a média da métrica *F-measure*, exatidão por aplicativo, para os dois algoritmos com o melhor compromisso (trade-off) entre maior acurácia geral e menor custo computacional, AP e NBG.

F-measure é a melhor maneira de visualizarmos os resultados para as métricas *Precision* e *Recall* juntas, para tráfego em tempo real (Kim et al., 2008). Um modelo perfeito alcança seu melhor *score* em 1 e o pior *score* em 0. Como essa métrica é uma média, teremos uma visão mais exata da eficiência do classificador por aplicação.

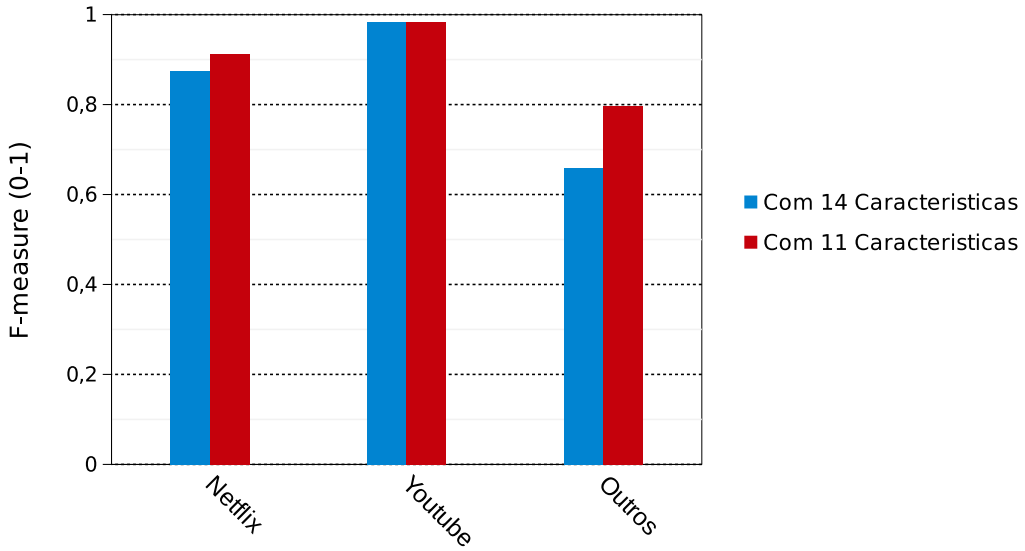


Figura 5.9: F-measure médio para o NBG com os dois grupos de características.

Para o algoritmo NBG, o módulo de classificação na arquitetura SDN obteve *scores* bem diferentes para cada aplicação. Com as 14 características, os valores alcançados para as aplicações com serviços de Netflix, YouTube e *Background* (Outros) foram de 0,87, 0,98 e 0,65, respectivamente. O score para a classificação de tráfego com aplicações de *Background* teve uma baixa eficiência, que afetará o balanceamento de carga proposto. Já para streaming de vídeo do tipo YouTube, foi obtido o melhor *score*. Com as 11

características, houve uma melhora geral dos *scores* para as classes de aplicação: 0,91, 0,98 e 0,79, respectivamente.

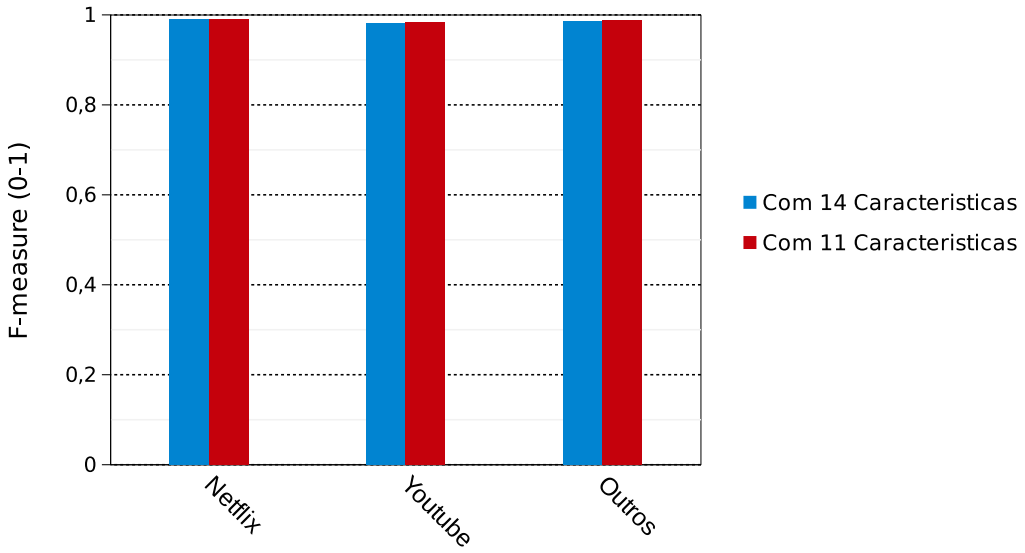


Figura 5.10: F-measure médio para o AP com os dois grupos de características.

Em contraste, o Algoritmo Proposto (AP) alcançou *scores* muito próximos de 1 para cada uma das classes de aplicação, Netflix, YouTube e *Background*, tanto com 14 quanto com 11 características: 0,99 e 0,99, Netflix, 0,98 e 0,99, YouTube, 0,98 e 0,99, *Background*, respectivamente. Destaque aqui não só para o alto valor obtido para cada classe, mas também para a uniformidade dos valores.

A Tabela 5.2 traz os resultados das métricas acurácia, custo computacional e *F-measure* para o Algoritmo Proposto (AP), demonstrando um melhor compromisso (trade-off) deste no cenário, com tráfego ativo na arquitetura SDN de rede local do Data Center.

Tabela 5.2: Resultados do compromisso das métricas do AP com 11 características.

Métricas		**Algoritmo Proposto (AP)**
Acurácia		98,60%
Custo Computacional		125 ms
F-measure	**Netflix**	0,99
	YouTube	0,99
	Background	0,99

Em suma, neste Cenário 1 foi analisado o desempenho do classificador proposto em uma arquitetura SDN com tráfego de streaming de vídeo em tempo real. O Algoritmo Proposto (AP) alcançou os melhores resultados, sinalizando um potencial para uso em monitoramento e tratamento de tráfego, aproveitando a visão global, dinâmica e programável do controlador SDN POX. Adicionalmente, foi possível identificar que os seis primeiros pacotes consecutivos de um fluxo desempenham um papel importante na classificação do tráfego de aplicativo em tempo real, em todos os modelos utilizados neste cenário.

5.4.3 Cenário 2 - Engenharia de Tráfego

Este cenário teve como objetivo produzir uma prova de conceito, capaz de demonstrar a atuação do classificador de tráfego desenvolvido neste trabalho em um ambiente de rede de Data Center. A arquitetura escolhida foi a apresentada na Figura 5.5, aplicando a Engenharia de Tráfego para balanceamento de carga entre servidores réplicas, com a divisão do tráfego de entrada de acordo com as três classes de aplicação e seus respectivos requisitos de prioridade de QoS (Tabela 4.2).

A Engenharia de Tráfego consiste em prover a adaptação do tráfego às condições da infraestrutura da rede, objetivando de forma integrada a simplicidade, bom desempenho para os usuários e uso eficiente dos recursos disponibilizados pela rede (Awduche et al., 2002). Com a Engenharia de Tráfego se torna factível o controle de importantes parâmetros dos fluxos, tais como a Qualidade de Serviço (*QoS - Quality of Service*) e o gerenciamento da vazão ativa (*traffic shaping*).

Na literatura, há diversas soluções que aplicam métodos de Engenharia de tráfego para melhoramento do roteamento (Fortz et al., 2002) (Bhatia et al., 2015) e balanceamento de carga (Wang et al., 2011) (Barabas et al., 2011) (Gao et al., 2017). Especificamente, o método de balanceamento de carga divide uma carga (demanda) por diversos recursos computacionais disponíveis na rede. Tal balanceamento pode ser implementado por destino, por fluxo ou por pacote (Augustin et al., 2007).

Quando ocorre um grande aumento do volume do tráfego de entrada em um Data Center, existe uma alta probabilidade de ser atingido o limiar de uso dos recursos computacionais de determinados enlaces da rede, enquanto outros caminhos poderão estar com seus recursos totalmente disponíveis (ociosos). Nos trechos comprometidos, a tendência é o significativo aumento das perdas e dos atrasos dos pacotes, com a lentidão se tornando ainda maior devido às sucessivas retransmissões, comprometendo as demandas mínimas de QoS e podendo levar ao travamento ou desconexão dos serviços por assinatura (clientes). O problema poderá ocorrer em momentos não determinísti-

cos, independente do total dos recursos computacionais projetados para o ambiente do Data Center.

Uma maneira de atenuar esta situação seria a adoção de técnicas de predição do tráfego, as quais se adéquam melhor a situações de fluxos de tráfego homogêneo com aplicações elásticas (Boutaba et al., 2018). Entretanto, neste trabalho a proposta foi a de tratar tráfego de rede de aplicações não homogêneas e não elásticas, com um limiar de tempo para a identificação do mesmo em torno de 100 ms ~ 150 ms (atraso fim a fim).

A solução adotada foi a de utilizar o classificador SDN desenvolvido, para que o tráfego seja dividido de modo equilibrado entre os caminhos. Não há o uso da análise de portas ou inspeção profunda dos pacotes. O classificador fornece uma base de conhecimento para determinar os níveis de desempenho exigidos pelas classes de aplicação. Usando o módulo de classificação, o controlador OpenFlow SDN faz em tempo real a alocação dos fluxos entre os enlaces, executando uma Engenharia de Tráfego mais precisa e robusta de alocação de recursos, pois usa a classificação de tráfego com granulação fina.

5.4.3.1 Estruturação do Cenário

Na arquitetura da Figura 5.5, o tráfego de entrada das requisições dos clientes é encaminhado para uma porta do switch de borda, para ser balanceada entre os N servidores réplica. Neste switch, uma porta especifica faz a ligação com o controlador SDN POX, possibilitando uma conexão segura com o controlador através do protocolo OpenFlow. Embora a arquitetura possa ser aplicada para balancear o tráfego em um cenário com N servidores réplica, um em cada enlace, aqui foi definido N = 3 com granularidade fina para a classificação adotada (Tabela 4.2).

Em uma arquitetura de Data Center o switch núcleo deverá sempre ter uma taxa de transferência maior que os switches de borda, em média 10 Gbps e 1 Gbps, respectivamente (Benson et al., 2010). Contudo, para tornar os resultados mais confiáveis, pacotes pequenos e taxa de transferência relativamente baixas foram utilizados, pois o desempenho tanto do Open vSwitch quanto do controlador OpenFlow no ambiente Mininet são afetados pelo sistema operacional, pela capacidade disponível do processador e pela quantidade de memória alocada. Consequentemente, a vazão entre o switch de borda e cada servidor réplica no experimento foi estipulada em 10 Mbps.

Para caracterizar o cenário foram levantados os valores de vazão e latência entre o switch de borda e cada servidor réplica, como se segue. Para vazão, os valores foram obtidos através da ferramenta iperf[9], gerando um fluxo bidirecional durante um

[9]https://iperf.fr/

intervalo de 60 s para cada servidor réplica. O iperf é uma ferramenta frequentemente usada em trabalhos relacionados com o paradigma das Redes Definidas por Software (Farhady et al., 2015) (Huang et al., 2017), que reúne em uma única aplicação o relatório da análise de várias métricas, como a capacidade máxima fim-a-fim a nível de transporte, o *jitter* e a perda de pacotes. A latência foi obtida através do envio de 1000 pacotes do elemento que gera o tráfego de entrada para cada servidor réplica, através da ferramenta ping. Em ambos os levantamentos, o controlador POX operou com o modo *learning switch* do protocolo OpenFlow, de forma que switch de borda se comportasse como um switch comum. Os resultados médios obtidos da vazão máxima e da latência são apresentados na Tabela 5.3.

Tabela 5.3: Vazão Média, Latência Média e Desvio Padrão.

	Ferramenta (iperf)	**ICMP (ping)**	
Conexão Switch - Servidor	Vazão Máxima Média (Mbps)	Latência Média (milissegundos)	Desvio Padrão
Servidor Réplica A	9,536	1,203	0,024
Servidor Réplica B	9,547	1,115	0,029
Servidor Réplica C	9,541	1,159	0,033

Foram realizados experimentos com três tipos diferentes de fluxos de tráfego, gerados e modelados conforme descrito a seguir.

(i) Serviços de streaming de vídeo, gerados com a utilização da ferramenta VLC[10] (reprodutor multimídia, de código aberto, multiplataforma, que reproduz a maioria dos codificadores de mídia e formatos de vídeo), hospedados nos servidores réplicas A e B (executando o VLC RTP Server) e com o cliente (gerador de requisições para tráfego de entrada) assinando este serviço (executando o VLC RTP Client). Foram gerados tráfegos com as características dos dois serviços de streaming mais populares, Netflix e YouTube, com formatos de vídeo de 720p e 1080p, respectivamente, taxa de quadros padrão 24, 25 ou 30.

(ii) Tráfego de fundo (background), de troca de arquivos, simulando vários clientes enviando requisições concorrentes a um servidor Web (servidor réplica C), gerado pela ferramenta httperf[11]. O httperf (Mosberger e Jin, 1998) é uma ferramenta construída para medir o desempenho de servidores Web, podendo utilizar o protocolo HTTP em suas versões 1.0 e 1.1, oferecendo uma variedade de geradores

[10]http://www.videolan.org/vlc/index.html
[11]https://github.com/httperf/httperf

de carga. Foi retratada uma carga típica Web 2.0, onde os usuários requisitam e recebem de servidores Web arquivos maiores que simples páginas, com características de arquivos texto e fotos, com aproximadamente 1 MB de dados (Jeon et al., 2012).

(iii) Tráfego de fundo TCP gerado através do Scapy, entre o cliente (gerador de requisições para tráfego de entrada) e o servidor réplica C. O Scapy é uma ferramenta de manipulação de pacotes de rede que provê classes para: enviar pacotes de diversos tipos de protocolos e comparar requisições e respostas.

Algumas pesquisas que emularam ambiente de Data Centers testaram seus algoritmos com um ambiente livre de carga (Lei et al., 2015; Gholami e Akbari, 2015; Zhang et al., 2014). Contudo, a literatura vem sinalizando para que os ambientes emulados descrevam o cenário mais próximo do mundo real (Karakus e Durresi, 2017). Para tornar o cenário mais próximo do ambiente de Data Center real e mais confiável, adicionalmente foi gerada uma carga de fundo (background) na rede, equivalente o envio de 15% da capacidade de cada enlace, ou seja, 1,5 Mbps entre o switch borda e cada servidor réplica. Isso foi feito reduzindo-se a capacidade de cada enlace em 15%, ou seja, para 8,5 Mbps.

Com a utilização da função de relatório do iperf, foi possível capturar a vazão (*throughput*) da rede (unidirecional e bidirecional), perda de pacotes (*packet-loss*) e latência dos experimentos.

No controlador OpenFlow SDN foi implementado um evento para que, na inicialização, a tabela de encaminhamento OpenFlow no switch seja zerada. A partir daí, quando o controlador recebia uma mensagem OpenFlow do tipo **Packet-In** do switch de borda (sinalizando um novo fluxo recebido), ele passava a mesma para o módulo de classificação. Este fazia o tratamento do tráfego, analisando os N (N=6) primeiros pacotes do fluxo, determinando a classe do mesmo e informando o controlador. Em sequência, o controlador enviava ao switch uma mensagem OpenFlow **FlowMod**, atualizando a sua tabela de encaminhamento com as informações de uma tupla, associando as respectivas decisões de balanceamento de carga à classe de cada fluxo de aplicação.

Considerando as taxas exigidas por diferentes resoluções de vídeo, a natureza do serviço como gratuita ou como assinatura paga e o tráfego de background, o tratamento do tráfego caracterizou os fluxos em três classes, de acordo com os requisitos de prioridade e de QoS. A Tabela 5.4 apresenta essas categorias.

Tabela 5.4: Categoria de Tráfego para Balanceamento.

Categoria A	Streaming de vídeo com requisitos de assinatura (*Netflix*)
Categoria B	Streaming de vídeo sem requisitos de assinatura (*YouTube*)
Categoria C	Tráfego de *background*

Para produzir a carga de tráfego, primeiro foram disparados os streamings de vídeo, cada um com taxa agregada de 5 Mbps. Nos momentos iniciais, os parâmetros de dados do tráfego de entrada foram registrados, tais como tamanho de dados, taxa de transferência e perda de pacotes. Após 23 s do início, o tráfego de fundo foi inserido, com carga agregada de 5 Mbps, gerando fluxos concorrentes. Cada um dos testes foi repetido 10 vezes, para gerar uma maior confiabilidade nos dados obtidos, gerando uma média das 10 repetições.

A Tabela 5.5 apresenta a descrição dos streaming que compõem o tráfego de cada categoria de vídeo.

Tabela 5.5: Descrição dos Streamings de Vídeo.

		Taxa de Codificação	**Resolução**	**Taxa Agregada (por categoria)**
Categoria A (Netflix)	streaming 1	1,5 Mbps	1080p	**5,0 Mbps**
	streaming 2	1,5 Mbps	1080p	
	streaming 3	2,0 Mbps	1080p	
Categoria B (YouTube)	streaming 1	0,5 Mbps	720p	**5,0 Mbps**
	streaming 2	0,5 Mbps	720p	
	streaming 3	1,5 Mbps	1080p	
	streaming 4	2,5 Mbps	1080p	

5.4.3.2 Execução dos Experimentos

Os experimentos objetivaram levantar os valores de vazão (*throughput*), taxa de perda de pacotes (*packet-loss*) e variação dos tempos de chegada de pacotes (*jitter*), nas situações **sem** e **com** a utilização da técnica de balanceamento de carga com uso do módulo de classificação de tráfego.

Na primeira situação, sem balanceamento, os tráfegos das três categorias (A, B e C) utilizaram apenas um dos enlaces, com um único servidor réplica. O objetivo foi o de sobrecarregar o enlace e levá-lo ao congestionamento, com a degradação da vazão, da perda de pacotes e do *jitter*.

Na segunda situação, com balanceamento, todos os fluxos são distribuídos para enlaces específicos, definidos de acordo com os requisitos mínimos de QoS para cada

categoria de fluxo. Para a prova de conceito, foi aplicada a distribuição dos serviços por servidor réplica (Al-Fares et al., 2008) (Benson et al., 2010) .

As Figuras 5.11 e 5.12 apresentam os resultados para a vazão de todos os tráfegos transmitidos, respectivamente nas situações "sem balanceamento"e "com balanceamento".

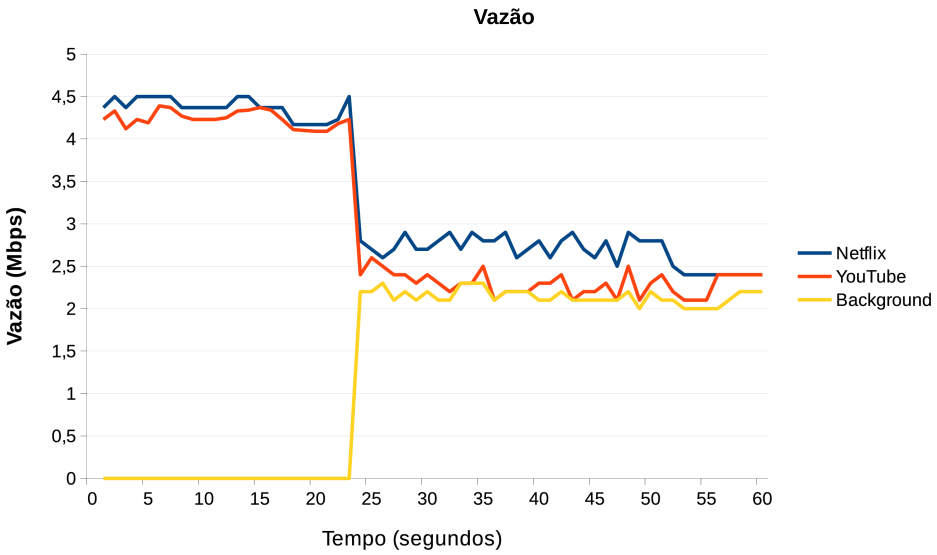


Figura 5.11: Vazão dos tráfegos (sem balanceamento).

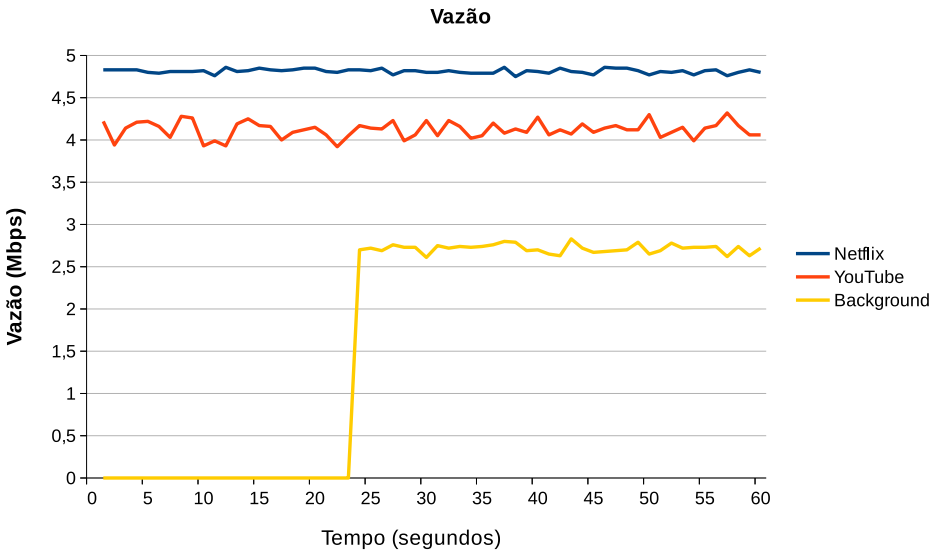


Figura 5.12: Vazão dos tráfegos (com balanceamento).

Na Figura 5.11 é possível notar que, inicialmente (0 s até cerca de 23 s), o streaming Netflix apresentou uma vazão média de 4,37 Mbps e o streaming YouTube apresentou

uma vazão média de 4,23 Mbps. Após 23 s, quando o tráfego de background se juntou à rede, os valores passaram para 2,28 Mbps, 2,45 Mbps e 2,20 Mbps, para Netflix, YouTube e Background, respectivamente.

Com o balanceamento de carga ativo, Figura 5.12, é possível observar que desde o início do experimento as vazões não sofrem degradações. Mesmo após cerca de 23 s, com o tráfego de background iniciado, o controlador SDN também conseguiu balancear o tráfego de acordo com os requisitos de QoS e as taxas de transferências são mantidas. A Tabela 5.6 resume os valores médios de vazão obtidos (para as situações **sem** e **com** balanceamento) após o tráfego de background ter se juntado aos fluxos.

Tabela 5.6: Aumento da vazão com a utilização de balanceamento de carga.

Métrica	**Categoria A (Netflix)**	**Categoria B (YouTube)**	**Categoria C (Background)**
Vazão (Sem balanceamento)	2,28 Mbps	2,45 Mbps	2,20 Mbps
Vazão (Com balanceamento)	4,81 Mbps	4,05 Mbps	2,58 Mbps

Além disso, as taxas de perda de pacotes, **sem** e **com** balanceamento, também foram capturadas, conforme mostrado nas Figuras 5.13 e 5.14, respectivamente.

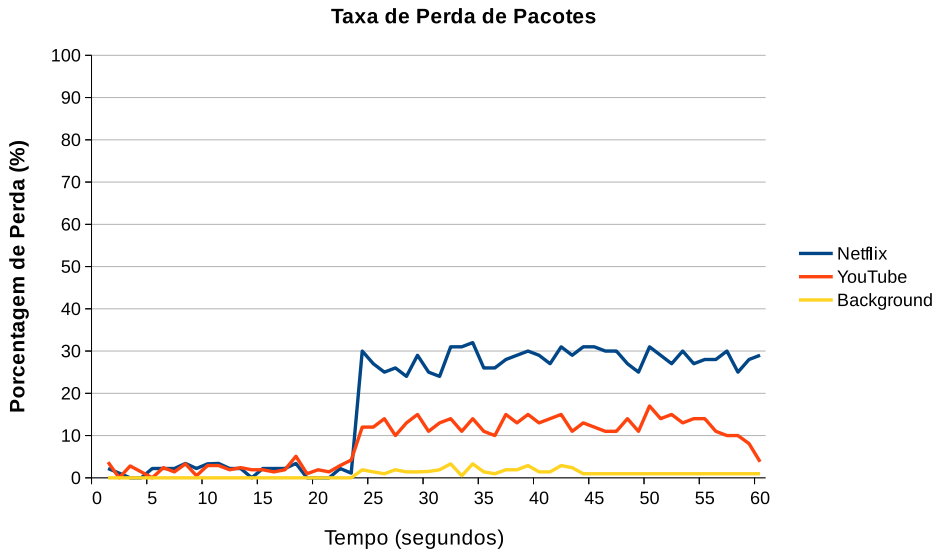


Figura 5.13: Taxa de perda de pacotes (sem balanceamento).

A Figura 5.13 apresenta um percentual de perda muito alto sem o uso de balanceamento, principalmente para tráfego de streaming, variando em torno de 1,1% a 31%

para a Categoria A (Netflix), 0,49% a 17% para a Categoria B (YouTube) e 0,48% a 3,3% para a Categoria C (Background).

Com balanceamento, o percentual de perda diminui significativamente para todas as categorias de tráfego, conforme apresentado na Figura 5.14.

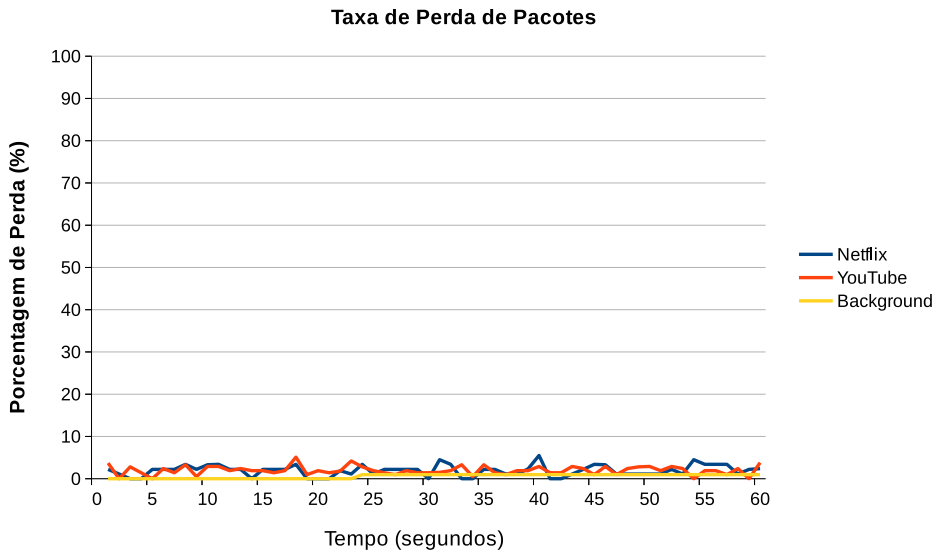


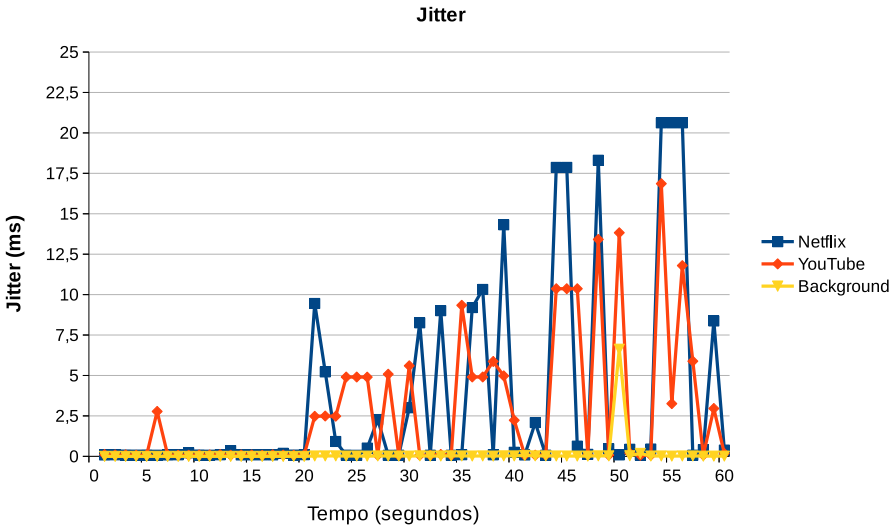
Figura 5.14: Taxa de perda de pacotes (com balanceamento).

A Tabela 5.7 resume os valores médios da porcentagem de perda (**sem** e **com** balanceamento), após o tráfego de Background ter se juntado aos fluxos. As categorias de tráfego de streaming de vídeo, aplicações não elásticas, foram as que mais sofreram com a entrada do tráfego de background, sem a aplicação do balanceamento de carga.

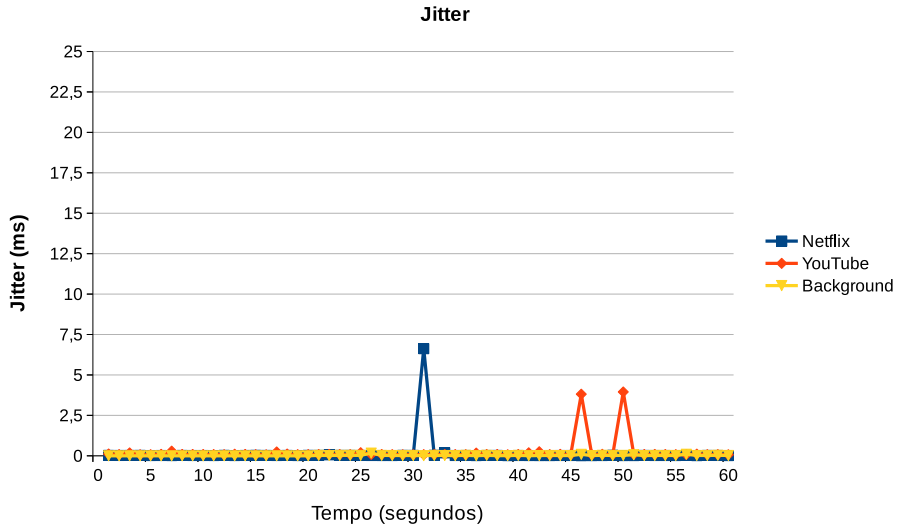
Tabela 5.7: Diminuição da Pocentagem de Perda de Pacotes com utilização a utilização de balanceamento.

Métrica	**Categoria A (Netflix)**	**Categoria B (YouTube)**	**Categoria C (Background)**
Porcentagem de Perdas (Sem balanceamento)	28,24%	12, 37%	1,43%
Porcentagem de Perdas (Com balanceamento)	1,91%	1,99%	0,74%

Também foram levantados os valores para a métrica *jitter*. O *jitter* é a variação (desvio padrão) dos tempos de chegada de pacotes, ou seja, o *jitter* pode ser considerado como a variação da latência (Peterson e Davie, 2007). As Figuras 5.15 (a) e 5.15(b) apresentam os valores do jitter durante a transmissão dos fluxos, nas situações **sem** balanceamento e **com** balanceamento, respectivamente.



(a) sem balanceamento



(b) com balanceamento

Figura 5.15: *Jitter* durante a transmissão

Pode ser observada na Figura 5.15(a) uma grande variação em determinados instantes dos valores para o *jitter*, especificamente para o tráfego não elástico. Para aplicações como transmissão de vídeo, não importa se pacotes demoram 10 ms, 20 ms ou 100 ms para chegarem ao receptor, contando que o tempo entre as transmissões de pacotes seja constante. Para o caso de streamings de vídeo, altas taxas de *jitter* podem degradar a transmissão. Com a utilização do balanceamento de carga, Figura 5.15(b), os valores de *jitter* de cada categoria de fluxo diminuem significativamente. Isso foi possível pois, ao balancear o tráfego para os respectivos enlaces e servidores réplicas, foi possível

manter o mínimo de taxa de transferência exigida por cada categoria.

5.5 Discussão dos Resultados

Este capítulo detalhou a implementação de um cenário de rede SDN em um Data Center com arquitetura em duas camadas, utilizando o emulador Mininet. O módulo de classificação de tráfego on-line foi acoplado ao controlador SDN POX, como proposta de uma nova abordagem para o balanceamento de carga do tráfego de entrada.

Inicialmente, no Cenário 1 foi testado o desempenho do módulo de classificação on-line SDN com tráfego ativo. O objetivo foi mostrar a viabilidade do módulo, com todas a extensões feitas no protocolo OpenFlow, para realizar a classificação de tráfego em tempo real com granularidade fina.

Os resultados mostraram que o módulo SDN de classificação de tráfego com o Algoritmo Proposto (AP) obteve os melhores resultados, obtendo uma acurácia média geral de 98,6%, contra 89,99% e 71,12%, dos outros algoritmos de ML testados, NBG e SVM, respectivamente. Ressalta-se que para todos os algoritmos ML a acurácia teve um incremento no seu valor após a análise dos N primeiros pacotes do tráfego, N variando de 1~10. O resultado da análise mostrou que os primeiros seis pacotes consecutivos são os mais representativos, para uma classificação de tráfego com características tanto não elásticas quanto elásticas. A partir do sétimo pacote a acurácia média diminuiu significativamente, confirmando que os primeiros seis pacotes são os mais relevantes para uma classificação precisa do tráfego.

Em seguida foi avaliado o custo computacional para a classificação. Os valores do custo computacional para todos os algoritmos ficou acima do limiar recomendável de 100 ms (Baldi e Ofek, 2000). O modelo com algoritmo AP obteve o menor custo, na ordem de 125 ms. Contudo, considerando o comportamento observado do switch SDN emulado, é esperado um melhor desempenho quando executado em switches SDN físicos.

Com o propósito de averiguar a classificação de tráfego com granularidade fina, foi avaliada a precisão na classificação para os dois melhores resultados no compromisso entre acurácia e custo computacional, AP e NBG, respectivamente. Este é um fator extremamente importante na classificação de tráfego on-line, pois alguns modelos podem apresentar boa precisão (*precision*), mas nem sempre uma sensibilidade (*recall*) elevada, sendo que o contrário também pode acontecer. Frente a esse fato, foi usada a métrica F-measure, que faz um compromisso entre essas duas medidas, atribuindo, inclusive o mesmo grau de importância para ambas. O AP gerou altos valores de *score*

para todas as classes de tráfego dos fluxos, comprovando alta precisão e sensibilidade para a classificação com granulação fina.

O Cenário 2 analisou o módulo de classificação de tráfego on-line em suporte ao balanceamento de carga, para a divisão de tráfego de entrada em um Data Center com rede SDN, como proposta inovadora de Engenharia de Tráfego. O cenário avaliou a atuação do classificador na melhoria da disponibilidade dos recursos de vazão e na utilização otimizada de servidores réplica.

Os resultados mostraram que o uso do módulo de classificação de tráfego no balanceamento de carga melhorou significativamente os valores da taxa de transferência e reduziu a perda de pacotes e os valores de *jitter*, quando combinado com o uso de vários servidores réplicas.

Neste cenário, forçando a situação de gargalo gerado pelo acúmulo dos tráfegos, foi possível observar os problemas advindos da não aplicação da técnica de balanceamento da carga nos fluxos. Quando a taxa de transferência cai para 2,5 Mbps, que é o mínimo suficiente para transmitir um vídeo em 720p, as aplicações não elásticas sofrem degradações nos valores das métricas, diminuindo assim a QoS oferecida. Sem uma política de balanceamento de carga, apenas um enlace é escolhido para o encaminhamento de todos as classes de tráfego entre cliente e servidor, explicando a baixa taxa disponível de vazão e a grande perda de pacotes.

É importante mencionar que a abordagem de se utilizar servidores réplicas para atender a vários serviços simultaneamente, com o devido tratamento de tráfego proporcionado pelo módulo de classificação, permite definir, configurar e alocar os recursos de enlace baseados nos requisitos de cada serviço, possibilitando a configuração dos recursos computacionais disponibilizados por cada servidor réplica, de maneira pré-definida e escalável.

Essa abordagem objetiva resolver o problema de gerenciamento da rede, quando se aplica técnicas tradicionais de tratamento do tráfego. Os aplicativos ou serviços emergentes podem não ser identificados pelos esquemas tradicionais, como simplesmente verificar endereços IP de origem/destino na camada de rede ou números de porta de origem/destino, bem como protocolos na camada de transporte, pois podem passar por diferentes esquemas de NAT (*Network Address Translation*) ou VPN (*Virtual Private Network*) ou não utilizar os números de porta padrão.

Por fim, os resultados indicam que o uso do módulo de classificação de tráfego online, integrado em um cenário de rede SDN e combinado com o recurso de servidores réplica, funciona como uma técnica inovadora de balanceamento de carga. Além disso, os resultados mostraram que o Algoritmo Proposto melhora significativamente o desempenho geral da rede, mostrando-se uma técnica promissora para ser aplicado

em tráfego em tempo real.

Capítulo 6

Conclusões

6.1 Considerações Gerais e Contribuições

Na atualidade, com o rápido desenvolvimento de dispositivos inteligentes e novas tecnologias de redes, o tráfego de dados no mundo está crescendo de forma extremamente rápida e se tornando mais complexo e heterogêneo. Para organizar, gerenciar, manter e otimizar com eficiência as redes em virtude desse aumento no tráfego, é necessário implantar mais inteligência em sua infraestrutura. Neste cenário, o aprendizado de máquina (ML - *Machine Learning*) tem se apresentado como solução em diferentes situações, permitindo a automação de diversos mecanismos. Essencialmente, isso se torna possível devido à explosão na disponibilidade de dados, às melhorias significativas nas técnicas de ML e ao avanço nas capacidades de computação.

Entretanto, ao se aplicar o ML é importante considerar as características da tarefa que estar sendo abordada. Para a classificação de tráfego em tempo real, em geral, as abordagens tradicionais disponíveis da literatura tendem a trabalhar com base no protocolo, produzindo uma classificação de granulação grossa e oferecendo um gerenciamento da rede de maneira inadequada para cada aplicação. Por exemplo, essas abordagens podem classificar fluxos de tráfego de streaming de vídeo, mas são incapazes de distinguir aplicativos de streaming específicos, como Netflix, YouTube e Vimeo. Isso acontece porque atualmente as aplicações da Internet tornaram-se mais sofisticadas e com mais e mais protocolos carregando diferentes aplicações. Outro aspecto é o fato das abordagens nem sempre focarem seus esforços na seleção do modelo com o melhor compromisso entre desempenho e custo computacional do classificador. Normalmente o foco é na discussão dos méritos algorítmicos do modelo de ML, ao invés de trabalharem nas questões relacionadas à análise do tempo despendido nas fases de treinamento e teste de cada modelo.

De modo complementar, um controlador SDN centralizado possui uma visão de rede global, o que facilita o controle e o gerenciamento. As técnicas de aprendizado de máquina podem trazer inteligência ao controlador SDN, permitindo análise de

dados, otimização de rede e fornecimento automatizado de serviços na mesma. Em outras palavras, a capacidade de aprendizado permite que o controlador SDN consiga autonomamente tomar decisões para se adaptar aos ambientes de rede. Contudo, embora a classificação de tráfego em ambiente SDN tenha obtido muitas conquistas, ainda permanecem vários desafios a serem resolvidos.

Este trabalho visou atender algumas dessas demandas, especialmente ao definir um modelo onde a classificação refinada contribui para analisar a composição da rede, entender o perfil do usuário e fornecer uma melhor QoS, mantendo o compromisso entre desempenho e custo computacional na classificação de tráfego em tempo real. Na elaboração do texto, a discussão dos méritos técnicos e teóricos foi apresentada, para subsidiar a avaliação do modelo proposto.

O objetivo aqui foi o desenvolvimento de um Classificador de Tráfego em tempo real para ambiente de Redes Definidas por Software (SDN), para aplicação em cenários de Engenharia de Tráfego em Data Centers. A principal contribuição foi a arquitetura de um módulo de classificação de tráfego multiclasse em tempo real. Foi utilizado um algoritmo com uma nova abordagem para o relaxamento da hipótese de independência entre os atributos do modelo Naive Bayes, pois sabe-se que na maioria dos casos a suposição de independência dos atributos de uma instância é falsa. A ideia central para o relaxamento foi garantir que a matriz de covariância seja positiva definida e simétrica, fazendo com que o algoritmo não mais assuma que a covariância entre os atributos seja igual a zero. Os resultados dos experimentos mostraram que essa abordagem obteve o melhor compromisso entre a acurácia média geral e o custo computacional, em relação aos outros modelos analisados.

Foi necessário usar um mecanismo de seleção de características com base nas correlações das variáveis de entrada, pois aplicações de streaming geralmente se comportam de maneira semelhante aos aplicativos de transferência de dados em massa, como o FTP (*File Transfer Protocol*). Para o método de seleção de características, foi utilizada a abordagem de remoção de características que possuem um alto valor de correlação com outras características, mas que não possuem uma correlação apreciável com sua classe. Isso pode ser feito estabelecendo um intervalo de corte para os valores de correlação entre as variáveis de entrada e um outro intervalo de corte para a correlação entre as variáveis de entrada e de classificação. Essa metodologia de seleção de característica se revelou crucial para o desempenho do algoritmo.

Outro problema enfrentado, recorrente na literatura de classificação de tráfego de vídeo, foi a indisponibilidade de conjunto de dados, em repositórios de aprendizado de máquina, contendo vídeos em streaming com características apenas do tráfego da rede. Para sanear este problema, foi preciso:

- a criação de uma base de dados com conteúdo de captura de tráfego de vídeo, além de scripts Bash Linux que podem ser automatizados para gerarem capturas periodicamente;
- o desenvolvimento e implementação de um script na linguagem Python, responsável por extrair os dados da ferramenta de captura;
- a análise da base de dados levantada.

Como prova de conceito, o módulo de classificação de tráfego em tempo real foi integrado ao ambiente SDN, como uma funcionalidade para o gerenciamento do tráfego. Para resolver a falta de suporte adequado para realizar tarefas relacionadas à classificação de tráfego, foi desenvolvida uma extensão dos contadores nativos do protocolo OpenFlow, através das características estatísticas obtidas pelo *script* extrator. Isso permitiu uma informação mais precisa sobre o comportamento dos fluxos dos tráfegos gerados em tempo real, possibilitando a classificação com granulação fina e tornando possível uma melhor abordagem para o problema de gerenciamento da rede, comparada ao tratamento do tráfego de maneira tradicional. Foram obtidos *scores* da métrica *F-measure* muito próximos de 1 (0,99 em valor) para cada uma das três classes de aplicação consideradas.

Um aspecto adicional importante considerado no desenvolvimento dessa arquitetura foi a possibilidade do classificador suportar a implementação de uma variedade de algoritmos de ML para classificação, sem grandes dificuldades de configuração. Em complemento, foi implementada a possibilidade do módulo de classificação carregar o modelo do classificador treinado periodicamente a partir de uma estrutura construída na arquitetura, que atualiza o modelo dinamicamente para a classificação em tempo real. Com isso, é possível tentar mitigar possíveis mudanças nas características do tráfego em tempo real (evolução do tráfego de rede e novos protocolos), treinando-o com um novo conjunto de dados periodicamente.

O aprendizado de máquina (ML) permite que o sistema examine os dados e deduza o conhecimento. Em essência, o ML identifica e explora padrões ocultos em dados de treinamento. Os padrões aprendidos são usados para analisar dados desconhecidos, de modo que possam ser agrupados ou mapeados para grupos conhecidos.

Para melhorar a estimativa ou precisão de classificação dos modelos treinados por ML, neste cenário de redes de computadores, são necessários conjuntos de dados de treinamento suficientes e descritivos. Porém, uma questão surgiu no desenvolvimento deste trabalho: o que são dados de treinamento suficientes e descritivos em redes? Esse questionamento também foi observado em grande parte do referencial teórico

abordado. Portanto, uma área de pesquisa ainda promissora para estudo é a relação entre o tamanho do conjunto de dados de treinamento, as características da rede e o desempenho dos modelos de ML. Isso foi comprovado nos experimentos, pois os progressos observados nos algoritmos de ML dependeram da disponibilidade do conjunto de dados de treinamentos gerados e normalizados com alta qualidade.

No entanto, é difícil obter amostras de fluxo de rede com anotações de alta qualidade em uma ampla gama de aplicativos, como streaming de vídeo. Para resolver este problema, uma possível abordagem é tornar públicos os conjuntos de dados de tráfego de rede utilizados na literatura, solução comum em vários aplicativos populares de aprendizado de máquina, tais como reconhecimento de imagem. É necessária uma iniciativa semelhante para a publicação de conjuntos de dados na área de redes, especialmente com relação aos tráfegos das camadas de enlace e rede.

Além disso, para tarefa de classificação de tráfego, é importante uma análise comportamental das características ao se aplicar um modelo de construção e seleção. O método de seleção de característica proposto foi modelado com uma análise prévia, considerando as classes de aplicações de streaming de interesse. Observou-se que as médias do tempo entre chegada dos pacotes, do comprimento do datagrama IP e do comprimento do quadro Ethernet, por exemplo, foram características não muito representativas para aplicações de streaming.

Para a tarefa de classificação descrita no trabalho, o método implementado de extração e seleção de características mostrou-se adequado. No entanto, algumas características podem sofrer distorções quando aplicadas em cenários de redes distintos. Por exemplo, o tamanho médio do pacote pode ser distorcido por fragmentação intermediária e criptografia, e os tempos de chegada entre pacotes e suas distribuições podem ser afetadas por filas nos roteadores. Portanto é imperativo considerar as classes de interesse, ou seja, aplicações e cenário topológico da rede, antes de selecionar as características para um problema específico de classificação de tráfego. Sendo assim, a extração e a seleção cuidadosa de características são tarefas cruciais para o desempenho dos modelos de ML.

Finalmente, também são necessários mais estudos sobre a viabilidade da classificação antecipada usando uma quantidade inicial de pacotes consecutivos de um fluxo. É extremamente importante identificar uma quantidade N de pacotes iniciais que contenham informações úteis para distinguir os tipos de tráfego nos fluxos. Para a tarefa deste trabalho, descobriu-se que as informações estatísticas dos seis primeiros pacotes consecutivos geraram maior precisão do que as dos demais pacotes.

6.2 Propostas de Continuidade

Com relação à metodologia proposta de classificação de tráfego e às considerações levantadas, sugere-se a continuidade do estudo e o aprimoramento da integração com outras metodologias. Dentre as mais importantes, pode-se citar:

- aplicar outras técnicas de correlação de fluxos, usando características estatísticas discretizadas baseadas no modelo Bag-of-Flow (BoF);
- estender o algoritmo proposto para descobrir aplicações em cenários *zero-day*, anteriormente desconhecidas em sistemas de classificação de tráfego;
- estender o módulo de classificação para executar retreinamentos com periodicidade adaptativa (em diferentes dias e horários) em novos conjuntos de dados, aplicando técnicas de Algoritmo Evolutivo para detectar mudanças nas características do tráfego de rede;
- implementar e configurar o módulo de classificação em uma rede SDN real, com switches OpenFlow e controladores físicos, sem a utilização do emulador Mininet e com outros controladores de código-fonte aberto como o ONOS e o Floodlight.

Referências Bibliográficas

Agarwal, S., Kodialam, M., e Lakshman, T. (2013). Traffic engineering in software defined networks. In *INFOCOM, 2013 Proceedings IEEE*, páginas 2211–2219. IEEE.

Aguado, A., Davis, M., Peng, S., Álvarez, M. V., López, V., Szyrkowiec, T., Autenrieth, A., Vilalta, R., Mayoral, A., Muñoz, R., et al. (2016). Dynamic virtual network reconfiguration over sdn orchestrated multitechnology optical transport domains. *Journal of Lightwave Technology*, 34(8):1933–1938.

Al-Fares, M., Loukissas, A., e Vahdat, A. (2008). A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, páginas 63–74. ACM.

Alshammari, R. e Zincir-Heywood, A. N. (2009). Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, páginas 1–8. IEEE.

Alshammari, R. e Zincir-Heywood, A. N. (2015). Identification of VoIP Encrypted Traffic using a Machine Learning Approach. *Journal of King Saud University-Computer and Information Sciences*, 27(1):77–92.

Amaral, P., Dinis, J., Pinto, P., Bernardo, L., Tavares, J., e Mamede, H. S. (2016). Machine learning in software defined networks: Data collection and traffic classification. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, páginas 1–5. IEEE.

Arslan, Z., Alemdaroglu, A., e Canberk, B. (2013). A traffic-aware controller design for next generation software defined networks. In *2013 First International Black Sea Conference on Communications and Networking (BlackSeaCom)*, páginas 167–171. IEEE.

Augustin, B., Friedman, T., e Teixeira, R. (2007). Measuring load-balanced paths in the internet. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, páginas 149–160. ACM.

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., e Xiao, X. (2002). Overview and principles of internet traffic engineering. Relatório técnico.

Baldi, M. e Ofek, Y. (2000). End-to-End Delay Analysis of Videoconferencing over Packet-Switched Networks. *IEEE/ACM Transactions On Networking*, 8(4):479–492.

Barabas, T., Ionescu, D., e Veres, S. (2011). Pim-ssm within diffserv-aware mpls traffic engineering. In *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, páginas 263–268. IEEE.

Barroso, L. A., Clidaras, J., e Hölzle, U. (2013). The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154.

Benson, T., Akella, A., e Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, páginas 267–280. ACM.

Bhatia, R., Hao, F., Kodialam, M., e Lakshman, T. (2015). Optimized network traffic engineering using segment routing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, páginas 657–665. IEEE.

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., e Weiss, W. (1998). An Architecture for Differentiated Services. Relatório técnico.

Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., e Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16.

Chandrashekar, G. e Sahin, F. (2014). A Survey on Feature Selection Methods. *Computers & Electrical Engineering*, 40(1):16–28.

Channegowda, M., Nejabati, R., e Simeonidou, D. (2013). Software-defined optical networks technology and infrastructure: Enabling software-defined optical network operations [invited]. *Journal of Optical Communications and Networking*, 5(10):A274–A282.

Chappell, L. e Combs, G. (2010). *Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide*. Protocol Analysis Institute, Chappell University.

Chen, X., Lyu, M. R., e King, I. (2017). Toward Efficient and Accurate Covariance Matrix Estimation on Compressed Data. In *International Conference on Machine Learning*, páginas 767–776.

Cisco, V. (2018). Cisco Visual Networking Index: Forecast and Trends, 2017–2022. *White Paper*.

Committee, O. M. E. et al. (2012). Software-defined networking: The new norm for networks. *ONF white paper*, páginas 1–12.

Cui, H., Zhu, Y., Yao, Y., Yufeng, L., e Liu, Y. (2014). Design of intelligent capabilities in sdn. In *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*, páginas 1–5. IEEE.

Cvijetic, N., Tanaka, A., Ji, P. N., Sethuraman, K., Murakami, S., e Wang, T. (2014). Sdn and openflow for dynamic flex-grid optical access and aggregation networks. *Journal of Lightwave Technology*, 32(4):864–870.

Da Silva, A. S., Machado, C. C., Bisol, R. V., Granville, L. Z., e Schaeffer-Filho, A. (2015). Identification and selection of flow features for accurate traffic classification in sdn. In *2015 IEEE 14th International Symposium on Network Computing and Applications*, páginas 134–141. IEEE.

Datta, J., Kataria, N., e Hubballi, N. (2015). Network Traffic Classification in Encrypted Environment: A Case Study of Google Hangout. In *Communications (NCC), 2015 Twenty First National Conference on*, páginas 1–6. IEEE.

D'Errico, J. (2013). Finding the Nearest Positive Definite Matrix. *Disponível em https://www.mathworks.com/matlabcentral/fileexchange/42885-nearestspd/. (Acessado em 20 de Dezembro de 2014).*

di Torino, P. (2019). Open datasets and libraries. Disponível em https://smartdata.polito.it/category/open-datasets/. Acessado em 13 Julho 2018.

Dong, H., Sun, G.-L., e Li, D.-D. (2013). A Hybrid Method for Network Traffic Classification. In *Measurement, Information and Control (ICMIC), 2013 International Conference on*, volume 1, páginas 653–656. IEEE.

Farhady, H., Lee, H., e Nakao, A. (2015). Software-defined networking: A survey. *Computer Networks*, 81:79–95.

Finsterbusch, M., Richter, C., Rocha, E., Muller, J.-A., e Hanssgen, K. (2014). A survey of payload-based traffic classification approaches. *IEEE Communications Surveys & Tutorials*, 16(2):1135–1156.

Fortz, B., Rexford, J., e Thorup, M. (2002). Traffic engineering with traditional ip routing protocols. *IEEE communications Magazine*, 40(10):118–124.

FOUNDATION, O. N. (2012). Software-Defined Networking: The New Norm for Networks. Disponível em https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. Acessado em 20 Fevereiro 2019.

FOUNDATION, O. N. (2013). Optical transport working group. Disponível em https://www.opennetworking.org/images/stories/downloads/working-groups/charter-optical-transport.pdf. Acessado em 20 Maio 2016.

Fu, Z., Liu, Z., e Li, J. (2017). Efficient parallelization of regular expression matching for deep inspection. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, páginas 1–9. IEEE.

Gao, X., Kong, L., Li, W., Liang, W., Chen, Y., e Chen, G. (2017). Traffic load balancing schemes for devolved controllers in mega data centers. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):572–585.

Gentle, J. E. (2012). Numerical Linear Algebra for Applications in Statistics. páginas 93–95. Springer Science & Business Media.

Gholami, M. e Akbari, B. (2015). Congestion control in software defined data center networks through flow rerouting. In *2015 23rd Iranian Conference on Electrical Engineering*, páginas 654–657. IEEE.

Golub, G. H. e Van Loan, C. F. (2012). *Matrix Computations*, volume 3. JHU Press.

Gomes, R. L. e Madeira, E. R. M. (2012). A Traffic Classification Agent for Virtual Networks Based on QoS Classes. *IEEE Latin America Transactions*, 10(3):1734–1741.

Goo, Y.-H., Shim, K.-S., Lee, S.-K., e Kim, M.-S. (2016). Payload signature structure for accurate application traffic classification. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, páginas 1–4. IEEE.

Gringeri, S., Bitar, N., e Xia, T. J. (2013). Extending software defined network principles to include optical transport. *IEEE Communications Magazine*, 51(3):32–40.

Gringoli, F., Salgarelli, L., Dusi, M., Cascarano, N., Risso, F., et al. (2009). Gt: picking up the truth from the ground for internet traffic. *ACM SIGCOMM Computer Communication Review*, 39(5):12–18.

Guedes, D., Vieira, L., Vieira, M., Rodrigues, H., e Nunes, R. V. (2012). Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, 30(4):160–210.

Hall, M. A. (1999). Correlation-Based Feature Selection for Machine Learning, Ph.D Thesis. University of Waikato Hamilton.

Hall, M. A. (2000). Correlation-Based Feature Selection of Discrete and Numeric Class Machine Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, páginas 359–366.

Han, J., Pei, J., e Kamber, M. (2011). *Data Mining: Concepts and Techniques*. Elsevier.

Higham, N. J. (1988). Computing a Nearest Symmetric Positive Semidefinite Matrix. *Linear Algebra and its Applications*, 103:103–118.

Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., e Wattenhofer, R. (2013). Achieving high utilization with software-driven wan. In *ACM SIGCOMM Computer Communication Review*, volume 43, páginas 15–26. ACM.

Hsu, C.-W. e Lin, C.-J. (2002). A Comparison of Methods for Multiclass Support Vector Machines. *IEEE transactions on Neural Networks*, 13(2):415–425.

Huang, T., Yu, F. R., Zhang, C., Liu, J., Zhang, J., e Liu, Y. (2017). A survey on large-scale software defined networking (sdn) testbeds: Approaches and challenges. *IEEE Communications Surveys & Tutorials*, 19(2):891–917.

IANA (2019). Service Name and Transport Protocol Port Number Registry. Disponível em: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml/. Acessado em 20 de Dezembro de 2018.

Inter-Datacenter, W. (2012). with centralized te using sdn and openflow.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., et al. (2013). B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14.

Jarschel, M., Wamser, F., Hohn, T., Zinner, T., e Tran-Gia, P. (2013). Sdn-based application-aware networking on the example of youtube video streaming. In *2013 Second European Workshop on Software Defined Networks*, páginas 87–92. IEEE.

Jeon, M., Kim, Y., Hwang, J., Lee, J., e Seo, E. (2012). Workload characterization and performance implications of large-scale blog servers. *ACM Transactions on the Web (TWEB)*, 6(4):16.

Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., e Matsuoka, S. (2009). Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. *Communications Magazine, IEEE*, 47(11):66–73.

John, G. H. e Langley, P. (1995). Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, páginas 338–345. Morgan Kaufmann Publishers Inc.

Jolliffe, I. T. (2002). *Principal Component Analysis. 2nd ed.* New York, NY: Springer.

Kaggle (2016). Your home for data science. Disponível em https://www.kaggle.com. Acessado em 20 Maio 2016.

Kaoprakhon, S. e Visoottiviseth, V. (2009). Classification of Audio and Video Traffic over HTTP Protocol. In *Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on*, páginas 1534–1539. IEEE.

Karakus, M. e Durresi, A. (2017). A survey: Control plane scalability issues and approaches in software-defined networking (sdn). *Computer Networks*, 112:279–293.

Kim, H., Claffy, K. C., Fomenkov, M., Barman, D., Faloutsos, M., e Lee, K. (2008). Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT conference*, página 11. ACM.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., e Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Krishnamurthy, A., Chandrabose, S. P., e Gember-Jacobson, A. (2014). Pratyaastha: An efficient elastic distributed sdn control plane. In *Proceedings of the third workshop on Hot topics in software defined networking*, páginas 133–138. ACM.

L7-Filter (2009). Application Layer Packet Classifier for Linux. Disponível em: http://l7-filter.sourceforge.net. Acessado em 20 de Dezembro de 2018.

Langley, P., Iba, W., e Thompson, K. (1992). An Analysis of Bayesian Classifiers. In *National Conference on Artificial Intelligence*, volume 90, páginas 223–228.

Lederer, S. (2015). Why Youtube & Netflix use MPEG-DASH in HTML5. *Disponível em https://bitmovin.com/status-mpeg-dash-today-youtube-netflix-use-html5-beyond/(Acessado em 20 de Dezembro de 2018).(February 2015)*.

Leduc, G., Abrahamsson, H., Balon, S., Bessler, S., D'Arienzo, M., Delcourt, O., Domingo-Pascual, J., Cerav-Erbas, S., Gojmerac, I., Masip, X., et al. (2006). An open source traffic engineering toolbox. *Computer Communications*, 29(5):593–610.

Lei, Y.-C., Wang, K., e Hsu, Y.-H. (2015). Multipath routing in sdn-based data center networks. In *2015 European Conference on Networks and Communications (EuCNC)*, páginas 365–369. IEEE.

Li, W., Abdin, K., Dann, R., e Moore, A. (2013). Approaching Real-Time Network Traffic Classification. Relatório técnico.

Li, Y. e Li, J. (2014). Multiclassifier: A combination of dpi and ml for application-layer classification in sdn. In *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, páginas 682–686. IEEE.

Lichman, M. (2015). Uci machine learning repository, 2013. *URL http://archive.ics.uci.edu/ml*, 114.

Linux (2018). cron(8) - linux man page. Disponível em https://linux.die.net/man/8/cron. Acessado em 20 Julho 2018.

Liu, L., De Vel, O., Han, Q.-L., Zhang, J., e Xiang, Y. (2018). Detecting and preventing cyber insider threats: a survey. *IEEE Communications Surveys & Tutorials*, 20(2):1397–1417.

Lopez-Martin, M., Carro, B., Lloret, J., Egea, S., e Sanchez-Esguevillas, A. (2018). Deep Learning Model for Multimedia Quality of Experience Prediction Based on Network Flow Packets. *IEEE Communications Magazine*, 56(9):110–117.

Madhukar, A. e Williamson, C. (2006). A longitudinal study of p2p traffic classification. In *null*, páginas 179–188. IEEE.

MATLAB (2016). *version 9.0.0 (R2016a)*. The MathWorks Inc., Natick, Massachusetts.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

McKeown, N., Parulkar, G., et al. (2010). Software defined networks and openflow.

Moore, A. W. e Papagiannaki, K. (2005). Toward the accurate identification of network applications. In *International Workshop on Passive and Active Network Measurement*, páginas 41–54. Springer.

Mosberger, D. e Jin, T. (1998). httperf?a tool for measuring web server performance. *ACM SIGMETRICS Performance Evaluation Review*, 26(3):31–37.

Nagaraj, K. e Katti, S. (2014). Procel: Smart traffic handling for a scalable software epc. In *Proceedings of the third workshop on Hot topics in software defined networking*, páginas 43–48. ACM.

Nascimento, Z., Sadok, D., Fernandes, S., e Kelner, J. (2014). Multi-Objective Optimization of a Hybrid Model for Network Traffic Classification by Combining Machine Learning Techniques. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, páginas 2116–2122. IEEE.

Netflix (2019). Netflix. Disponível em https://help.netflix.com/pt/node/306. Acessado em 13 Julho 2018.

Ng, B., Hayes, M., e Seah, W. K. (2015). Developing a traffic classification platform for enterprise networks with sdn: Experiences & lessons learned. In *2015 IFIP Networking Conference (IFIP Networking)*, páginas 1–9. IEEE.

Nguyen, T. T. e Armitage, G. (2006). Training on Multiple Sub-Flows to Optimise the Use of Machine Learning Classifiers in Real-World IP Networks. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, páginas 369–376. IEEE.

Nguyen, T. T. e Armitage, G. (2008). Clustering to Assist Supervised Machine Learning for Real-Time IP Traffic Classification. In *Communications, 2008. ICC'08. IEEE International Conference on*, páginas 5857–5862. IEEE.

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., e Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634.

OpenDPI (2012). Disponível em: https://github.com/thomasbhatia/OpenDPI. Acessado em 20 de Dezembro de 2018.

Pan, S. J. e Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Parsaei, M. R., Sobouti, M. J., Khayami, S. R., e Javidan, R. (2017). Network traffic classification using machine learning techniques over software defined networks. *International Journal Of Advanced Computer Science And Applications*, 8(7):220–225.

Peng, L., Yang, B., e Chen, Y. (2015). Effective packet number for early stage internet traffic identification. *Neurocomputing*, 156:252–267.

Peterson, L. L. e Davie, B. S. (2007). *Computer Networks: A Systems Approach*. Elsevier.

Pfaff, B., Lantz, B., Heller, B., Barker, C., Beckmann, C., et al. (2012). Openflow switch specification v1. 3.0. *Open Netw. Found., Menlo Park, CA, USA, Tech. Rep. ONF TS-006*.

Qazi, Z. A., Lee, J., Jin, T., Bellala, G., Arndt, M., e Noubir, G. (2013). Application-awareness in sdn. In *ACM SIGCOMM computer communication review*, volume 43, páginas 487–488. ACM.

Qilin, M. e Weikang, S. (2015). A load balancing method based on sdn. In *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, páginas 18–21. IEEE.

Rao, A., Legout, A., Lim, Y.-s., Towsley, D., Barakat, C., e Dabbous, W. (2011a). Network Characteristics of Video Streaming Traffic. In *Proceedings of the Seventh Conference on Emerging Networking Experiments and Technologies*, páginas 1–12. ACM.

Rao, A., Legout, A., Lim, Y.-s., Towsley, D., Barakat, C., e Dabbous, W. (2011b). Network Characteristics of Video Streaming Traffic. http://www-sop.inria.fr/members/Arnaud.Legout/Projects/streaming.html. Acessado em 20 de Dezembro de 2018.

Rish, I. (2001). An Empirical Study of the Naive Bayes Classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, páginas 41–46. IBM.

Rothenberg, C. E., Nascimento, M. R., Salvador, M. R., e Magalhães, M. F. (2010). Openflow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Cad. CPqD Tecnologia, Campinas*, 7(1):65–76.

Sammut, C. e Webb, G. I. (2011). *Encyclopedia of Machine Learning*. Springer Science & Business Media.

Savas, S. S., Ma, C., Tornatore, M., e Mukherjee, B. (2016). Backup reprovisioning with partial protection for disaster-survivable software-defined optical networks. *Photonic Network Communications*, 31(2):186–195.

Sen, S., Spatscheck, O., e Wang, D. (2004). Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*, páginas 512–521. ACM.

Senthil Ganesh, N. e Ranjani, S. (2015). Dynamic load balancing using software defined networks. *International Journal of Computer Applications*, páginas 0975–8887.

Sharma, P., Banerjee, S., Tandel, S., Aguiar, R., Amorim, R., e Pinheiro, D. (2013). Enhancing network management frameworks with sdn-like control. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, páginas 688–691. IEEE.

Shen, G., Guo, H., e Bose, S. K. (2016). Survivable elastic optical networks: survey and perspective. *Photonic Network Communications*, 31(1):71–87.

Shi, H., Li, H., Zhang, D., Cheng, C., e Wu, W. (2017). Efficient and Robust Feature Extraction and Selection for Traffic Classification. *Computer Networks*, 119:1–16.

Shi, Y. e Biswas, S. (2016). Protocol-Independent Identification of Encrypted Video Traffic Sources using Traffic Analysis. In *Communications (ICC), 2016 IEEE International Conference on*, páginas 1–6. IEEE.

Shirali-Shahreza, S. e Ganjali, Y. (2013). Flexam: flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, páginas 167–168. ACM.

Singh, K., Agrawal, S., e Sohi, B. (2013). A Near Real-Time IP Traffic Classification Using Machine Learning. *International Journal of Intelligent Systems and Applications*, 5(3):83.

Sone, Y., Watanabe, A., Imajuku, W., Tsukishima, Y., Kozicki, B., Takara, H., e Jinno, M. (2011). Bandwidth squeezed restoration in spectrum-sliced elastic optical path networks (slice). *Journal of Optical Communications and Networking*, 3(3):223–233.

Song, D. (2015). Read the docs. Disponível em https://dpkt.readthedocs.io/en/latest/. Acessado em 13 Julho 2018.

Sun, G., Chen, T., Su, Y., e Li, C. (2018a). Internet Traffic Classification Based on Incremental Support Vector Machines. *Mobile Networks and Applications*, páginas 1–8.

Sun, G., Liang, L., Chen, T., Xiao, F., e Lang, F. (2018b). Network Traffic Classification Based on Transfer Learning. *Computers & Electrical Engineering*, páginas 1–8.

Sun, N., Zhang, J., Rimba, P., Gao, S., Xiang, Y., e Zhang, L. Y. (2018c). Data-driven Cybersecurity incident prediction: A survey. *IEEE Communications Surveys & Tutorials, DOI: 10.1109/COMST.2018.2885561.*

Syed, N. A., Liu, H., e Sung, K. K. (1999). Handling Concept Drifts in Incremental Learning with Support Vector Machines. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 317–321. ACM.

Tongaonkar, A., Torres, R., Iliofotou, M., Keralapura, R., e Nucci, A. (2015). Towards Self Adaptive Network Traffic Classification. *Computer Communications*, 56:35–46.

Ubik, S. e Žejdl, P. (2010). Evaluating Application-Layer Classification Using a Machine Learning Technique over Different High Speed Networks. In *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*, páginas 387–391. IEEE.

UniBS, T. T. N. G. . (2011). The ground truth software tools. Disponível em http://netweb.ing.unibs.it/~ntw/. Acessado em 13 Julho 2018.

Voyant, C., Notton, G., Kalogirou, S., Nivet, M.-L., Paoli, C., Motte, F., e Fouilloy, A. (2017). Machine Learning Methods for Solar Radiation Forecasting: A Review. *Renewable Energy*, 105:569–582.

Wang, N., Fagear, A., e Pavlou, G. (2011). Adaptive post-failure load balancing in fast reroute enabled ip networks. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, páginas 470–477. IEEE.

Wang, P., Lin, S.-C., e Luo, M. (2016). A framework for qos-aware traffic classification using semi-supervised machine learning in sdns. In *2016 IEEE International Conference on Services Computing (SCC)*, páginas 760–765. IEEE.

Wang, Y. e Yu, S.-Z. (2009). Supervised Learning Real-Time Traffic Classifiers. *Journal of Networks*, 4(7):622–629.

Wickboldt, J. A., De Jesus, W. P., Isolani, P. H., Both, C. B., Rochol, J., e Granville, L. Z. (2015). Software-defined networking: management requirements and challenges. *IEEE Communications Magazine*, 53(1):278–285.

Xiao, P., Liu, N., Li, Y., Lu, Y., Tang, X.-j., Wang, H.-W., e Li, M.-X. (2016). A traffic classification method with spectral clustering in sdn. In *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, páginas 391–394. IEEE.

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., e Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(1):393–430.

Yamansavascilar, B., Guvensan, M. A., Yavuz, A. G., e Karsligil, M. E. (2017). Application Identification via Network Traffic Classification. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*, páginas 843–848. IEEE.

Yan, J. e Yuan, J. (2018). A Survey of Traffic Classification in Software Defined Networks. In *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, páginas 200–206. IEEE.

Yao, Q., Yang, H., Zhu, R., Yu, A., Bai, W., Tan, Y., Zhang, J., e Xiao, H. (2018). Core, Mode, and Spectrum Assignment Based on Machine Learning in Space Division Multiplexing Elastic Optical Networks. *IEEE ACCESS*, 6:15898–15907.

Yu, A., Yang, H., Bai, W., He, L., Xiao, H., e Zhang, J. (2018). Leveraging Deep Learning to Achieve Efficient Resource Allocation with Traffic Evaluation in Datacenter Optical Networks. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, páginas 1–3. IEEE.

Yu, X., Zhao, Y., Zhang, J., Wang, X., Wang, J., e Zhang, J. (2014). Spectrum engineering in flexible grid data center optical networks. *Optical Switching and Networking*, 14:282–288.

Zhang, H., Guo, X., Yan, J., Liu, B., e Shuai, Q. (2014). Sdn-based ecmp algorithm for data center networks. In *2014 IEEE Computers, Communications and IT Applications Conference*, páginas 13–18. IEEE.

Zhang, J., Chen, C., Xiang, Y., Zhou, W., e Xiang, Y. (2013a). Internet traffic classification by aggregating correlated naive bayes predictions. *IEEE transactions on information forensics and security*, 8(1):5–15.

Zhang, J., Chen, X., Xiang, Y., Zhou, W., e Wu, J. (2015). Robust Network traffic classification. *IEEE/ACM Transactions on Networking (TON)*, 23(4):1257–1270.

Zhang, J., Qian, Z., Shou, G., e Hu, Y. (2009). An Automated On-Line Traffic Flow Classification Scheme. In *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on*, páginas 1181–1184. IEEE.

Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., e Guan, Y. (2013b). Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed systems*, 24(1):104–117.

Zhang, J., Xiang, Y., Zhou, W., e Wang, Y. (2013c). Unsupervised traffic classification using flow statistical properties and ip packet payload. *Journal of Computer and System Sciences*, 79(5):573–585.

Zhang, J., Zhang, J., Zhao, Y., Yang, H., Yu, X., Wang, L., e Fu, X. (2013d). Experimental demonstration of openflow-based control plane for elastic lightpath provisioning in flexi-grid optical networks. *Optics express*, 21(2):1364–1373.

Zhu, H., Fan, H., Luo, X., e Jin, Y. (2015). Intelligent timeout master: Dynamic timeout for sdn-based data centers. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, páginas 734–737. IEEE.