UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

GOVINDA MOHINI GONZALEZ BEZERRA

Avaliação de segurança e desempenho de rádio definido por software em ambientes virtualizados para redes de acesso na Internet das Coisas

NITERÓI

2023

UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

GOVINDA MOHINI GONZALEZ BEZERRA

Avaliação de segurança e desempenho de rádio definido por software em ambientes virtualizados para redes de acesso na Internet das Coisas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações.

Orientadores:

Diogo Menezes Ferrazani Mattos
Tadeu Nagashima Ferreira

NITERÓI

2023

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

B574a Bezerra, GOVINDA MOHINI GONZALEZ
Avaliação de segurança e desempenho de rádio definido por software em ambientes virtualizados para redes de acesso na Internet das Coisas / GOVINDA MOHINI GONZALEZ Bezerra. - 2023.
97 p.: il.

Orientador: Diogo Menezes Ferrazani Mattos.
Coorientador: Tadeu Nagashima Ferreira.
Dissertação (mestrado)-Universidade Federal Fluminense, Escola de Engenharia, Niterói, 2023.

1. Rádio. 2. Avaliação de Desempenho. 3. Rede de Computadores. 4. Aprendizado de Máquina. 5. Produção intelectual. I. Mattos, Diogo Menezes Ferrazani, orientador. II. Ferreira, Tadeu Nagashima, coorientador. III. Universidade Federal Fluminense. Escola de Engenharia.IV. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

GOVINDA MOHINI GONZALEZ BEZERRA

Avaliação de segurança e desempenho de rádio definido por software em ambientes virtualizados para redes de acesso na Internet das Coisas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações.

Aprovada em 14 de junho de 2023.

BANCA EXAMINADORA

Prof. Tadeu N. Ferreira, D.Sc. – Orientador, UFF

Prof. Diogo M. F. Mattos, D.Sc. – Orientador, UFF

Profª. Dianne Scherly Varela de Medeiros, D.Sc. – UFF

Prof. Rodrigo de Souza Couto, D.Sc. – UFRJ

Niterói

2023

À minha família.

Agradecimentos

À minha família por todo apoio e carinho de sempre. Agradeço ao Lyno por toda a ajuda e companheirismo durante esse processo.

Aos amigos que fiz no laboratório MídiaCom pelos momentos de descontração que tornaram essa experiência muito mais divertida.

Aos Orientadores Diogo M. F. Mattos e Tadeu N. Ferreira, por todas as orientações, além de todo tempo dispendido nas reuniões de orientação e nas revisões dos trabalhos. Ao orientador Leonardo Guimarães pelas orientações e compreensão. Agradeço também aos professores Dianne S. V. Medeiros e Rodrigo S. Couto pela presença na banca examinadora.

À Marinha do Brasil pela oportunidade de realizar este trabalho.

Por fim, agradeço aos órgãos de fomento CNPq, CAPES, RNP e FAPERJ.

Resumo

A 5ª Geração (5G) de comunicação móvel é um salto na evolução na comunicação celular, oferecendo velocidades ultra-rápidas, comunicação massiva do tipo máquina e latência mínima. Essa tecnologia contribui para a conectividade ubíqua de pessoas e dispositivos, criando uma base sólida para a Internet das Coisas (IoT), realidade aumentada, inteligência artificial e outras tecnologias emergentes. Esta dissertação aborda dois aspectos relevantes para as redes 5G e está dividida em duas partes: o uso de dispositivos Rádio Definido por Software (SDR) em ambientes virtualizados para implementação de Rede de Acesso de Rádio (RAN) de redes móveis e a privacidade dos usuários na utilização de dispositivos de Internet das Coisas (IoT).

Os equipamentos de *hardware* programáveis SDR trazem grande flexibilidade aos projetos, uma vez que é possível realizar modificações significativas nas aplicações e ainda executá-las no mesmo dispositivo. Nesse contexto de flexibilidade e otimização, a virtualização oferece muitos benefícios em melhorar as taxas de utilização do *hardware*, reduzindo a ociosidade dos equipamentos. Assim, esta dissertação apresenta uma discussão detalhada dos desafios enfrentados ao implementar aplicações utilizando SDRs em ambientes virtualizados. Para tal, utilizam-se diferentes cenários de teste contendo duas diferentes formas de virtualização, máquinas virtuais e contêineres, e dois modelos diferentes de SDR do tipo *Universal Software Radio Peripheral* (USRP). Os resultados mostram que o contêiner tem um desempenho próximo ao sistema operacional nativo, enquanto as máquinas virtuais geram uma sobrecarga de processamento que afeta a qualidade das comunicações das aplicações SDR.

Na segunda parte desta dissertação, os aspectos abordados da rede 5G são a segurança e a privacidade de dispositivos de IoT. Cada vez mais, há uma maior variedade de dispositivos com preços acessíveis e, por consequência, seu uso está cada vez mais ubíquo em automações de rotinas em residências, empresas e indústrias. Conhecer os tipos de dispositivos IoT conectados em uma rede tem o potencial de revelar padrões de comportamentos do usuário e permitir a exploração de brechas de segurança dos dispositivos. Dessa forma, propõe-se o uso de três modelos de aprendizado de máquina, Rede Neural Convolucional (CNN), memória de curto e longo prazo - *Long Short-Term Memory* (LSTM) e *Extreme Gradient Boosting* (XGBoost), para inferir quais os tipos de dispositivos estão conectados em uma rede. Os resultados mostram que é possível descobrir os dispositivos através da análise estatística do tráfego de rede com acurácia, precisão e *recall* de 99%. Dessa forma, identifica-se o comprometimento da privacidade dos usuários, uma vez que categorias e modelos de equipamentos são revelados, indicando preferência e padrões de comportamento dos usuários.

Palavras-chave: 5G, virtualização, SDR, IoT, privacidade.

Abstract

The 5th Generation (5G) mobile communication is a leap forward in the evolution of cellular communication, offering increased speeds, massive machine-type communications (mMTC), and minimal latency. This technology contributes to the ubiquitous connectivity of people and devices, creating a solid foundation for the Internet of Things (IoT), augmented reality, artificial intelligence, and other emerging technologies. This work addresses two relevant aspects for 5G networks and is divided into two parts: the use of Software Defined Radio (SDR) devices in virtualized environments for the implementation of Radio Access Network (RAN) of mobile networks and the privacy of users of Internet of Things (IoT) devices.

SDR programmable hardware equipment brings great flexibility to projects, while making it possible to perform significant application modifications and run them on the same device. In this context of flexibility and optimization, virtualization offers many benefits in improving hardware utilization rates and reducing equipment idleness. Thus, this dissertation presents a detailed discussion of the challenges faced, when applications using SDRs are implemented in virtualized environments. For this purpose, different test scenarios containing two different forms of virtualization, virtual machines and containers, and two different Universal Software Radio Peripheral (USRP) SDR models are used. The results show that the container performs close to the native operating system, while the virtual machines cause a processing overhead that affects the quality of communications of SDR applications.

In the second part of this dissertation, 5G network aspects, such as, the security and privacy of IoT devices, are the main focus. There is an increasing variety of affordable devices, and, as a result, their use is becoming ubiquitous in routine automation in homes, businesses, and industries. Knowledge of the types of IoT devices connected to a network has the potential to reveal user behavior patterns and to allow the exploitation of device security vulnerabilities. Thus, this proposal uses three machine learning models, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Extreme Gradient Boosting (XGBoost), to infer which models of devices are connected on a network. The results show that it is possible to discover devices through statistical network traffic analysis with accuracy, precision, and recall of 99%. In this way, the compromise of users' privacy is identified since categories and models of equipment are revealed, indicating the preferences and behavior patterns of users.

Keywords: 5G, virtualization, SDR, IoT, privacy.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Sumário

Capítulo 1

Introdução

As redes celulares evoluíram ao longo do tempo através de diferentes gerações de tecnologia. A 1ª Geração (1G) foi introduzida em 1979 e permitia apenas chamadas de voz analógicas. Em seguida, a 2ª Geração (2G) surgiu na década de 1990 e introduziu a tecnologia digital, o que resultou em melhor qualidade de voz e a possibilidade de envio de mensagens curtas de texto. A 3ª Geração (3G), iniciada na década de 1990, trouxe uma melhoria significativa em termos de taxa de dados, viabilizando a Internet móvel e permitindo descarga de arquivos com taxas de até 2 Mb/s. Também nessa geração houve um aprimoramento em segurança que introduziu a autenticação mútua de elementos na rede, evitando conexões com estações-base falsas. A 4ª Geração (4G) oferece velocidades de dados de até 1 Gb/s e serviços sobre IP, como telefonia e TV. Além disso, o 4G introduziu novas tecnologias como transmissão com Múltiplas Entradas e Múltiplas Saídas, *Multiple-input and multiple-output* (MIMO), e Multiplexação por Divisão de Frequências Ortogonais (OFDM) que permitem o aumento da capacidade de transmissão e as altas taxas de transferência.

A 5ª Geração fornece velocidades ultra-rápidas de dados, latência muito baixa e maior capacidade de rede. As redes 5G têm o potencial de aprimorar uma ampla gama de setores, como transporte, saúde, indústria e entretenimento, visto que há uma crescente demanda por conectividade de banda larga confiável para viabilizar o desenvolvimento de tecnologias como carros autônomos, cirurgias remotas, realidade virtual e IoT. Essa crescente demanda por conectividade ubíqua estimula a implantação da infraestrutura de rede 5G em um ritmo acelerado. Enquanto apenas 10% dos Provedores de Serviços de Comunicações, *Communications Service Provider*s (CSPs), comercializaram serviços 5G em 2020, a expectativa é que esse número cresça para 60% até 2024 [1].

Os principais casos de uso que são favorecidos pela rede 5G são agrupados em: banda

larga aprimorada, *Enhanced Mobile Broadband* (eMBB), como vídeos de alta resolução e computação em nuvem; comunicações ultraconfiáveis de baixa latência, *Ultra-Reliable Low Latency Communications* (URLLC), como veículos autônomos e cirurgias remotas; e comunicações massivas entre máquinas, *Massive Machine Type Communications* (mMTC), que engloba a conexão de dispositivos IoT e automações residenciais e industriais. Essa diversidade de aplicações da rede 5G, cada qual com requisitos específicos de latência e de largura de banda, trazem novos desafios às comunicações da rede celular em termos de infraestrutura, segurança e privacidade.

Para atender às demandas desses diferentes cenários de uso, é necessário que as redes móveis sejam flexíveis. Tradicionalmente, uma RAN é constituída de unidades monolíticas provenientes de um grupo restrito de fabricantes, resultando em caixas pretas com reconfigurabilidade limitada e difícil coordenação entre os nós. Uma RAN aberta, ou *Open*-RAN (O-RAN), vislumbra a utilização de interfaces padronizadas e abertas, permitindo a interoperabilidade entre vendedores e estimulando a competição e inovação no setor. Outro fator importante e inovador das redes 5G, é a arquitetura baseada em serviços, *Service-Based Architecture* (SBA), que decompõe a rede em funções de rede isoladas que interagem entre si através de interfaces bem definidas. Essa arquitetura permite desacoplar o *hardware* do *software*, permitindo o uso de *hardware* de diferentes fabricantes, assim como a utilização de técnicas de virtualização e computação em nuvem para o processamento das funções de rede.

Nesse contexto de desacoplamento entre *hardware* e *software*, a tecnologia SDR pode ser empregada para trazer mais flexibilidade e adaptabilidade nas RANs das redes móveis, facilitando a prototipagem, testes e implantação de projetos de rede. O SDR consiste em uma tecnologia de comunicação por rádio totalmente programável que pode ser personalizada para atender a uma grande variedade de requisitos. Essa característica permite que os SDRs sejam usados em uma ampla gama de aplicações, incluindo estações-base de rede celular e redes de acesso de banda larga. Devido à variedade de dispositivos SDR comerciais, *Commercial Off-The-Shelf* (COTS), disponíveis, o emprego dessa tecnologia tem o potencial de reduzir significativamente o custo de construção e manutenção de redes celulares, ao mesmo tempo em que oferece uma grande flexibilidade de implantação. Embora o SDR tenha muitas vantagens potenciais, também existem vários desafios associados à implantação dessa tecnologia no ambiente virtualizado de uma RAN.

Outros desafios emergentes da ubiquidade de conexão proporcionada pela rede 5G são a segurança e privacidade dos dados. A disseminação de dispositivos IoT e a sua

incorporação cada vez maior nas atividades cotidianas tornaram esses dispositivos alvos de diversos ataques de segurança, cujo crescimento foi mais de 100% no primeiro semestre de 2021 [2]. Os dispositivos de IoT apresentam características de consumo de energia, processamento, uso de rede e armazenamento que diferem significativamente dos nós tradicionais da Internet, como servidores, computadores e notebooks [3]. Devido à diversidade de aplicações e de projetos, dispositivos IoT podem ser heterogêneos, dificultando uma abordagem universal para assegurar a segurança da conexão e dos dados. Assim, as questões de segurança e privacidade devem ser tratadas considerando restrições e características dos dispositivos IoT e as soluções devem ser propostas considerando também esse contexto. Além disso, pressionados pela demanda por dispositivos simples e de baixo custo, o projeto dos fabricantes tende a negligenciar medidas de segurança. Como consequência, os dispositivos IoT manifestam capacidade restrita para resistir a ataques cibernéticos [4].

1.1 Objetivos

Esta dissertação aborda dois temas principais relacionados a redes 5G. O primeiro consiste no uso de dispositivos SDR para implementação da RAN. Nesse contexto, é apresentada uma discussão detalhada dos vários desafios que devem ser enfrentados ao implementar uma RAN baseada em SDR e são discutidas possíveis soluções que têm potencial para superar esses desafios. Também é realizada uma análise dos principais parâmetros de desempenho que devem ser considerados ao avaliar o desempenho de uma RAN usando a tecnologia SDR.

O segundo tema desta dissertação é a segurança e a privacidade de dispositivos de IoT e o estudo de como técnicas de aprendizado de máquina podem ser empregadas para revelar informações sobre os dispositivos e, consequentemente, sobre os seus usuários. Cada equipamento IoT tem suas próprias características de comunicação, como protocolos utilizados, tamanhos de pacotes e servidores aos quais se conecta. A avaliação realizada neste trabalho visa determinar se as características estatísticas de comunicação são suficientes para identificar satisfatoriamente os dispositivos IoT que originaram a comunicação. Também é realizada uma análise de diferentes algoritmos de aprendizado de máquina com o intuito de verificar o modelo mais adequado para este uso.

1.2 Contribuições

Esta dissertação possui duas contribuições principais. A primeira é a verificação da viabilidade do uso de infraestruturas de rede móvel mais flexíveis e de menor custo. O estudo realizado analisa o impacto na comunicação de aplicações baseadas em *software* quando executadas em máquinas virtuais e contêineres. Paralelamente, a segunda contribuição desta dissertação é a representação estatística dos dados de tráfego de rede dos dispositivos IoT que permite a identificação precisa e acurada desses dispositivos por algoritmos de aprendizado de máquina. A terceira contribuição tem um caráter de alerta e conscientização para o uso dessas novas tecnologias, uma vez que a identificação dos dispositivos através dos seu tráfego de rede tem o potencial de comprometer a privacidade dos seus usuários.

1.3 Organização da dissertação

A dissertação está organizada da seguinte forma. O Capítulo 2 apresenta um panorama geral das redes 5G e dos dispositivos SDR. É descrita a arquitetura 5G, a RAN e os seus componentes, assim como são detalhados os princípios dos rádios SDR e da ferramenta GNU Radio. O Capítulo 3 analisa o uso de dispositivos virtualizados em ambientes SDR e apresenta os resultados obtidos. O Capítulo 4 discorre sobre questões de segurança nos dispositivos IoT e sobre o uso de algoritmos de aprendizado de máquina para a identificação dos dispositivos a partir da análise do tráfego de rede. O Capítulo 5 apresenta o conjunto de dados utilizado, os modelos empregados e os resultados obtidos. O Capítulo 6 conclui este trabalho e apresenta as propostas para trabalhos futuros.

Capítulo 2

Softwarização, Rede 5G e SDR

Atualmente, há um grande esforço dos projetistas de sistemas móveis em otimizar o uso de recursos e evitar o seu desperdício, seja com aquisição ou ociosidade de equipamentos, assim como gastos com energia elétrica. Algumas tecnologias que têm a capacidade de aumentar a taxa de utilização dos recursos computacionais e torná-los mais flexíveis têm sido estudadas para as redes de comunicações móveis das próximas gerações. Dessa forma, há uma tendência em definir as aplicações por *software*, utilizando *hardware* com multipropósitos e não-especializados. Com essa abordagem, é possível aproveitar um mesmo equipamento e utilizá-lo com diversas finalidades e, em alguns casos, de forma concomitante e multiusuários.

A implementação de um sistema de comunicações pode ser realizada de diferentes formas. A primeira, mais antiga e mais tradicional, é o desenvolvimento de um *hardware* especializado que realize todas e somente as funções do sistema em questão. A segunda, cada vez mais utilizada, é a utilização de um *hardware* de uso geral ou COTS para a implementação da funcionalidade via *software*. Um exemplo simples dessa dualidade é a utilização de um aparelho telefônico para efetuar uma ligação, o qual consiste em um *hardware* desenvolvido especificamente para essa finalidade, ou utilizar um programa em um computador, como o Skype, para realizar a ligação. Ambas as abordagens possuem suas vantagens e desvantagens. Em relação à primeira abordagem, a vantagem reside na velocidade com que a aplicação é executada. Uma vez que o *hardware* é projetado com um propósito específico, há uma otimização para a execução dessa aplicação. Por outro lado, o *hardware* é limitado ao escopo do projeto inicial, sendo custosa a expansão de novas funcionalidades ou adoção de novos padrões e tecnologias, uma vez que seria necessário substituir alguns componentes e/ou atualização do *firmware*. A segunda abordagem apresenta as vantagens e desvantagens contrárias à primeira, pois o uso de equipamentos

de uso geral permite uma alta reconfigurabilidade da aplicação executada, permitindo atualizações e a adoção de novas aplicações. Em contrapartida, o *hardware* de uso geral pode inserir novas camadas no sistema como, por exemplo, um sistema operacional, aumentando a sobrecarga de processamento e influenciando o tempo de execução e a latência de uma aplicação de comunicação. Uma terceira alternativa para a implementação de um sistema é utilizar uma abordagem híbrida das duas arquiteturas anteriores como, por exemplo, implementando uma parte crítica em latência por *hardware* e o restante da aplicação via *software*. Em resumo, a escolha de uma ou outra abordagem depende da relação custo-benefício entre versatilidade e velocidade de execução exigida pela aplicação a ser implementada.

A evolução e o consequente aumento da capacidade dos dispositivos de uso geral reduziram os efeitos negativos das implementações via *software*, uma vez que são capazes de atender os requisitos de diversos sistemas, mesmo sem a especialização dos componentes de *hardware*. Assim, os benefícios dessa abordagem já superam os pontos negativos em diversas áreas e pode-se observar um fenômeno chamado “softwarização", que consiste na troca do *hardware* especializado por implementações via *software*. Outro termo difundido, principalmente na área de rede e tecnologia da informação, é *Software-Defined Everything* (SDE), que, em tradução livre, significa tudo definido por *software*.

Em uma rede de computadores sem fio, existem três grupos básicos de equipamentos. O primeiro deles é o dispositivo de rádio que realiza a comunicação sem fio com outros dispositivos que transmitem e recebem dados. O segundo consiste nos elementos de rede fixa que encaminham os pacotes até o respectivo destino, como roteadores. Por último, estão os computadores pessoais e servidores que geram e recebem tráfego. A “softwarização", em cada uma dessas etapas, possui diferentes tecnologias. Em relação aos dispositivos de rádio, a técnica é chamada de SDR, já para o controle do roteamento do tráfego de rede, é possível utilizar uma Rede Definida por Software (SDN).

2.1 Virtualização

Virtualização é um conceito no qual se permite criar, via *software*, um ambiente imitando as características de um ambiente real, permitindo o compartilhamento do *hardware* entre múltiplas instâncias virtuais[5]. A virtualização permite consolidar a infraestrutura de computadores, otimizando o uso do *hardware* e reduzindo custos em equipamentos e eletricidade. Existem diversas técnicas de virtualização disponíveis com variados níveis

de abstração. O escopo deste trabalho abrange duas formas de virtualização: virtualização completa com Máquina Virtual (VM) e contêiner.

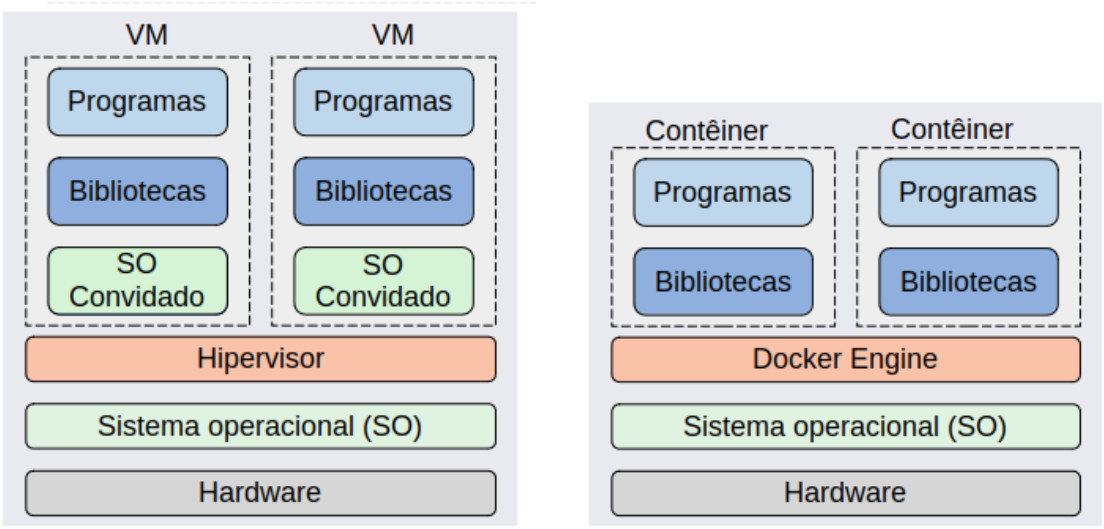
Na abordagem de VM, há uma camada de *software* chamada hipervisor, ou Virtual Machine Monitor (VMM), que cria e executa máquinas virtuais, permitindo que um computador hospedeiro suporte várias VMs. Há dois tipos de hipervisores: o tipo 1 é executado diretamente no *hardware*, enquanto o hipervisor do tipo 2 é executado como um programa de um Sistema Operacional (OS). Uma VM é uma emulação de um computador físico que permite executar um OS (*guest OS*) independente do OS hospedeiro (*host OS*). Esse desacoplamento entre *guest OS* e *host OS* permite que as VMs sejam movidas entre diferentes servidores, facilitando a portabilidade dos sistemas.

A segunda técnica utilizada neste trabalho usa uma abstração de contêiner que consiste em um ambiente contendo todas as ferramentas de sistema, bibliotecas e configurações para que um aplicativo seja executado de forma rápida, uniforme e independente da infraestrutura. Os contêineres compartilham o *kernel* do *host* OS, eliminando a necessidade de um *guest* OS completo, aumentando, assim, a eficiência e reduzindo custos dos servidores. O contêiner é uma unidade padrão de *software* que pode ser utilizada para distribuir e testar uma aplicação. Uma imagem é um modelo com instruções para a criação de um contêiner. Dessa forma, um contêiner é uma instância da imagem em tempo de execução, assim como um objeto é uma instância de uma classe na programação orientada a objeto. As imagens podem ser criadas com base em imagens pré-existentes e, a partir delas, podem ser adicionadas outras camadas. Por exemplo, a partir de uma imagem Ubuntu, é possível instalar programas, como GNURadio e UHD, e gerar uma nova imagem.

A plataforma de virtualização utilizada neste trabalho para criação de VMs é o VirtualBox [1] que possui uma interface de usuário simples, o que torna a ferramenta fácil de instalar, configurar e utilizar. O VirtualBox é *open-source*, gratuito e compatível com os OS Windows, Linux, Solaris e Mac. O VirtualBox utiliza um hipervisor tipo 2 e possui um pacote de expansão, chamado *VirtualBox Extension Pack*, que inclui suporte para USB2 e USB3. Também é possível realizar *snapshots* que permitem salvar o estado de uma VM e restaurá-la posteriormente.

A plataforma de contêiner utilizada neste trabalho é o Docker, que consiste em uma plataforma para desenvolvimento e execução de aplicativos. O Docker fornece um serviço

[1] Disponível em https://www.virtualbox.org/.



(a) Máquina virtual com hipervisor tipo 2.

(b) Contêiner.

Figura 2.1: Diferença entre máquinas virtuais e contêineres: (a) há um hipervisor e um *guest* OS para cada máquina virtual, enquanto em (b) há apenas uma camada adicional do Docker Engine.

de repositório de imagens, chamado Docker Hub [2], que contém uma grande variedade de imagens oriundas de vendedores de *software*, projetos *open-source* e da própria comunidade de usuários. O Docker disponibiliza imagens oficiais contendo sistema operacionais e aplicações populares que servem como ponto de partida para a maioria das imagens.

Em resumo sobre as formas de virtualização, a abordagem de contêiner emprega uma virtualização no nível de OS, compartilhando o *host OS* e acrescentando o mínimo necessário para a execução das aplicações. Enquanto as VMs utilizam a virtualização no nível de *hardware* e requerem a implantação de um *guest OS* completo. A Figura 2.1 ilustra a arquitetura desses dois tipos de virtualização abordados.

Os benefícios da "softwarização" são potencializados quando essa é utilizada concomitantemente com a virtualização, pois, enquanto a "softwarização" possibilita a definição de uma função via *software* e execução em *hardware* não-especializado, a virtualização permite o compartilhamento dos recursos disponíveis entre diversas aplicações. Apesar da técnica de virtualização de função estar muito difundida através de Virtualização das Funções de Rede (NFV) e SDN, a virtualização de funções de rádio ainda é embrionária e apresenta desafios que ainda necessitam ser estudados e avaliados, como processamento em tempo real e requisitos de latência. Programas de controle de fluxo de rádio, como o GNURadio, podem ser executados em ambientes virtualizados, porém não é possível

[2]Disponível em https://hub.docker.com/.

assumir que as aplicações serão virtualizadas sem sofrer um impacto da camada de virtualização. Também não é possível ignorar o impacto da "softwarização" em comparação à execução das aplicações em *hardware* especializados. Assim, é importante a execução de testes e avaliações para que o impacto da virtualização seja adequadamente mapeado.

2.2 Redes 5G

O *3rd Generation Partnership Project* (3GPP) é um projeto de um comitê de padronizações iniciado em 1998 para elaborar especificações e relatórios técnicos para a 3ª Geração de redes móveis celulares. O escopo do projeto evoluiu para a manutenção e desenvolvimento das especificações das gerações posteriores à 3G, tornando o 3GPP também responsável por redes 4G e 5G [6]. Devido à constante evolução das redes móveis, há um cuidado na elaboração de novos padrões para maximizar a compatibilidade com as tecnologias precedentes, de forma a manter a operabilidade dos equipamentos dos usuários e facilitar a implantação das novas tecnologias [6]. O 3GPP possui parceiros organizacionais (*Organizational Partners*), representados por sete organizações normatizadoras oriundas da Ásia, Europa e América do Norte, responsáveis por traduzir a documentação elaborada pelo 3GPP em normas aplicáveis no seu país ou região. O 3GPP também possui parceiros representantes do mercado (*Market Representation Partner*), constituídos de empresas convidadas pelos parceiros organizacionais para fornecer um ponto de vista de mercado em relação aos trabalhos desenvolvidos. Os trabalhos de especificação são realizados pelos grupos de especificação técnica, *Technical Specification Group* (TSG), cada qual em uma área de atuação específica, como núcleo de rede ou rede de acesso de rádio.

A 5ª Geração das redes móveis atende a uma grande variedade de necessidades de diferentes mercados, como o setor automobilístico, agrário e de saúde [7]. Para cada caso de uso, são definidos indicadores de desempenho, *Key Performance Indicator*s (KPIs) e requisitos de rede que respaldam o projeto da rede. Alguns dos principais KPIs utilizados são: disponibilidade e confiabilidade da rede, latência, mobilidade e perda de pacotes. A rede 5G se apoia no fatiamento de rede fim-a-fim, *End-to-End* (E2E), para atender os casos de uso previstos, como alta largura de banda, baixa latência e conexão massiva de dispositivos IoT. Uma fatia de rede possui um ciclo de vida que se inicia no projeto de uma rede com determinadas características, como área de cobertura, acordo de nível de serviço - *Service Level Agreement* (SLA), KPIs e equipamentos de usuário que são utilizados na rede. Uma vez criada, a fatia de rede precisa ser monitorada e gerenciada para garantir que o SLA contratado está sendo cumprido e que os recursos alocados são

compatíveis com a carga existente. Caso contrário, a fatia deverá ser reconfigurada de forma a cumprir os critérios do acordo.

O 3GPP definiu a rede 5G como uma SBA, que consiste em um conjunto de Função de Redes (NFs) interconectadas para construírem todas as funcionalidades da rede. Cada função fornece serviços específicos na rede, aos quais outras NFs podem ter acesso. A abordagem SBA contribui para a maior modularidade e reusabilidade dos sistemas em comparação ao modelo tradicional baseado em elementos da rede. Na rede 5G, os componentes são projetados utilizando o paradigma NFV. Dessa forma, as funções podem ser implementadas em uma infraestrutura virtualizada gerenciada por um orquestrador, tornando a rede extensível e adaptável. O fatiamento de rede permite a execução de múltiplas instâncias lógicas da rede móvel em uma infraestrutura compartilhada, combinando os SLAs exigidos, a volatilidade da demanda por serviços e a infraestrutura de maneira automatizada.

As redes móveis são compostas pela RAN e pelo núcleo da rede, *Core Network* (CN). No caso da rede 5G, o 3GPP especificou uma nova RAN, chamada *Next-Generation RAN* (NG-RAN), e um novo núcleo, o 5G *Core* (5GC). O projeto do 5GC utiliza tecnologia de virtualização e o paradigma de nuvem, porém, em termos de virtualização da rede de acesso de rádio, há muitas questões ainda em aberto e as soluções atuais não atendem a todos os desafios de isolamento, de alocação dinâmica de recursos e de heterogeneidade de *hardware*. Ainda assim, o emprego da virtualização de rádios é interessante, pois permite que diversos rádios virtuais, cada um com suas próprias características, coexistam em um mesmo *hardware* [8].

Uma outra característica das redes 5G é o conceito de redes não-públicas, *Non-Public Network*s (NPNs), que provêm os serviços de rede móvel nas dependências de uma empresa ou organização. As NPNs podem ser isoladas ou implantadas conjuntamente com redes públicas, porém, em ambos os casos, as redes públicas e privadas são dissociadas, o que garante a independência nas operações das redes. Em cenários de integração dessas duas redes, um agregador de serviços de rede supervisiona os recursos necessários para implementar essa integração. Seguindo a tendência de softwarização, existem projetos em desenvolvimento para implementação de redes celulares completamente funcionais utilizando apenas dispositivos comerciais SDR e *software* aberto, como, por exemplo, OpenAirInterface[3] e o projeto srsRAN [4].

[3]Disponível em https://openairinterface.org/.
[4]Disponível em https://www.srslte.com/.

Este trabalho se concentra na RAN das redes móveis e apresenta a seguir um breve histórico de sua evolução e de suas configurações, assim como algumas tendências de tecnologias relacionadas.

2.2.1 RAN

A RAN da rede móvel celular passou por diversas arquiteturas ao longo de suas gerações. Na rede 2G, o subsistema da Estação-Base, *Base Station Subsystem* (BSS), era composto de dois elementos, a estação-base transceptora, *Base Transceiver Station* (BTS), e o controlador da Estação Base, *Base Station controller* (BSC). A BTS era responsável pelos transceptores que se comunicavam com os telefones celulares e o BSC gerenciava a alocação dos canais de rádio e a coordenação de intercâmbio de rádio (*handover*) na área do BSS. Nessa geração, a BTS era composta da unidade de Radiofrequência (RF) e da unidade de banda base integradas em gabinetes. As redes móveis evoluíram para a 3ª Geração, com o BTS e o BSC, sendo substituídos pelo *Node* B (NB) e pelo *Radio Network Controller* (RNC), conforme ilustrado na Figura 2.2(a).

O NB das redes 3G evoluiu para o eNB nas redes 4G que incorpora as funcionalidades de controle, realizadas pelo RNC na rede anterior. Assim, cada eNB é responsável por gerenciar os seus recursos de rádio e a mobilidade na célula, desempenhando funções como modulação, codificação de canal, programação de recursos de rádio e coordenação de *handover*. Os eNB são conectados entre si por meio da interface X_2, de forma a suportar funções de coordenação e *handover* intercélulas, e se conectam ao núcleo da rede, *Evolved Packet Core* (EPC), através da interface S_1 para funções adicionais, como autenticação e roteamento de pacotes. A arquitetura da rede 4G está representado na Figura 2.2(b).

Nas redes 4G, além da configuração monolítica com as unidades de RF e banda base integradas, coexistiam eNBs onde as unidades eram separadas, sendo a unidade de banda base, *Baseband Unit* (BBU), mantida em gabinetes, enquanto as unidades de RF, *Remote Radio Unit* (RRU), eram instaladas em torres, conectadas ao BBU através de fibra ótica via *Common Public Radio Interface* (CRPI). Essa arquitetura é a RAN distribuída, *Distributed*-RAN (D-RAN), em contraponto ao modelo tradicional, no qual RRU e BBU eram integradas. Posteriormente, surgiu a arquitetura de RAN centralizada que agrupava os BBUs, formando BBUs *pools*, com o intuito de centralizar o gerenciamento e reduzir custos de manutenção. Uma outra configuração dessa arquitetura é a RAN na nuvem, *Cloud*-RAN (C-RAN), na qual os BBUs *pools* estão localizados na nuvem.

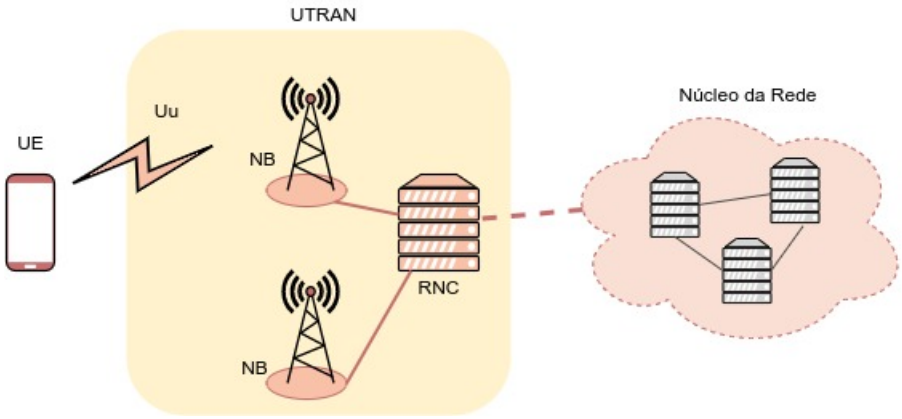
A arquitetura 5G, ilustrada na Figura 2.2(c), adota uma abordagem mais descentra-

lizada, com a separação das funções de controle de rádio e processamento de pacotes. A arquitetura 5G é projetada com conceitos de virtualização e nuvem. Assim, o gNB pode ser implementado como unidades distribuídas e virtualizadas, permitindo maior flexibilidade e escalabilidade.

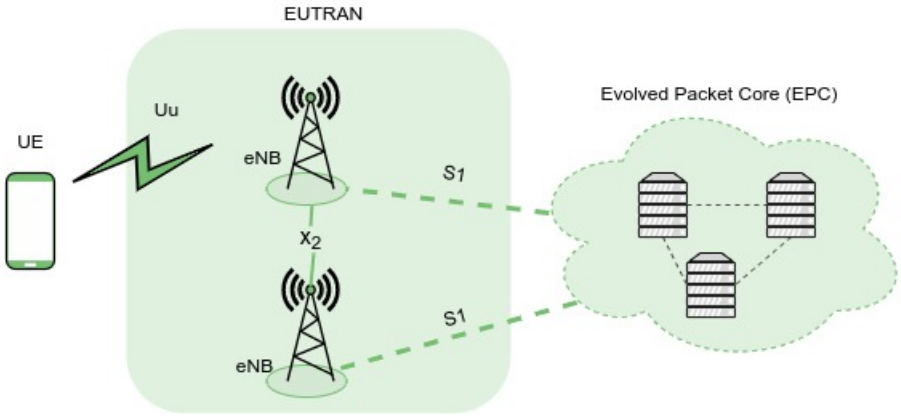
A arquitetura da rede 5G descrita emprega somente elementos da NG-RAN, sendo chamada de *Standalone* (SA). Para facilitar a transição entre as redes 4 e 5G, foram especificadas outras opções de implantações intermediárias que vislumbram a utilização de componentes da RAN 4G em conjunto com componentes da rede 5G, chamado de *Non-Standalone* (NSA) [9]. A arquitetura NSA ilustrada na Figura 2.3(a), representa uma das opções NSA definidas pelo 3GPP, onde a NG-RAN é utilizada em conjunto com a *Evolved-UMTS Terrestrial Radio Access Network* (E-UTRAN) e com o EPC, ambos legados da rede 4G. Nessa configuração, somente os serviços 4G estão disponíveis, porém os usuários se beneficiam de alguns aperfeiçoamentos da rede 5G, como baixa latência.

O principal elemento da NG-RAN, é o gNB que pode ser dividido em uma Unidade Centralizada (CU), gNB-CU, e uma ou mais Unidade Distribuída (DU), gNB-DU, conectados entre si através de uma interface F1. A Figura 2.4 ilustra as diferenças entre os eNB e gNB. É possível observar que o elemento BBU das redes 4G foi substituído pelas DU e CU. O 5GC se conecta ao CU através do enlace *backhaul* utilizando a interface NG definida pelo 3GPP, assim como a ligação entre CU e DU é realizada pela interface padronizada F_1, através do enlace *midhaul*. O 3GPP não padronizou o elemento Unidade de Rádio (RU), a sua interface com a DU ou o enlace *fronthaul*, ficando a cargo dos fabricantes a especificação desses elementos.

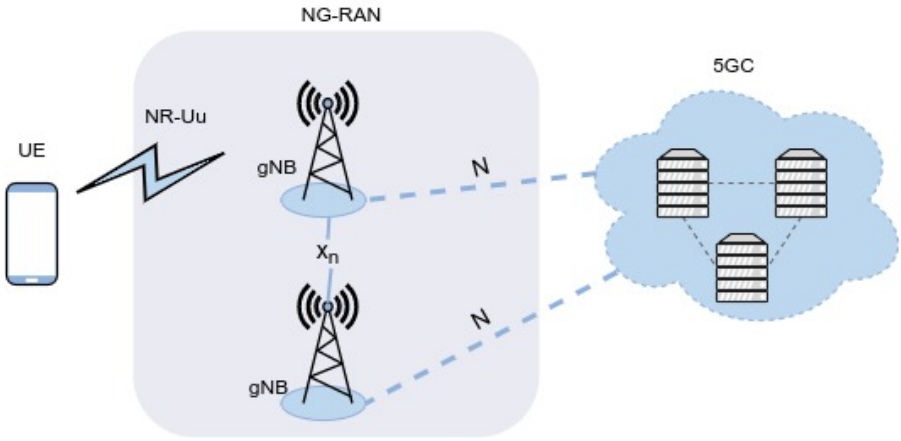
Devido à divisão do gNB em CU e DU, é necessário definir as funções de cada um dos blocos e a divisão da pilha de protocolos de comunicação entre as duas unidades. A Figura 2.5 mostra a pilha de protocolo do plano de controle da RAN 5G. O protocolo de Controle de Recurso de Rádio, *Radio Resource Control* (RRC), é utilizado para a configuração e controle das funções de rádio dos UE. O Protocolo de Convergência de Pacote de Dados, *Packet Data Convergence Protocol* (PDCP), implementa funcionalidades de segurança, compressão de cabeçalho e é responsável pela entrega E2E dos pacotes entre os UE e gNB. A camada de Controle de Enlace de Rádio, *Radio Link Control* (RLC), realiza correção de erros, fragmentação e reconstrução de pacotes. A camada de Controle de Acesso ao Meio, *Medium Access Control* (MAC), realiza a multiplexação/demultiplexação de dados em blocos de transportes para serem enviados/recebidos pela camada física. A camada física é responsável pela codificação do canal e modulação do sinal.



(a) Arquitetura 3G.

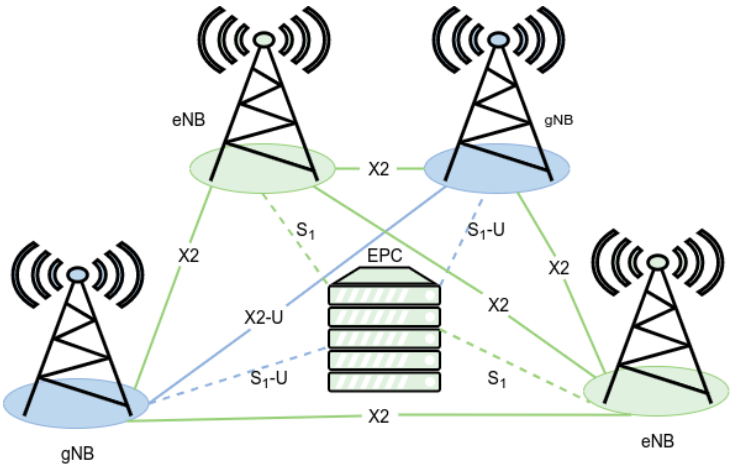


(b) Arquitetura 4G.



(c) Arquitetura 5G *Stand-Alone*.

Figura 2.2: Comparação das arquiteturas das redes 3G, 4G e 5G. (a) A RAN 3G é composta de NBs e RNC. (b) Na RAN 4G, além da interface com os equipamentos dos usuários, *User Equipment* (UE), o eNB assume funções de controle que, na geração anterior, era realizada pelo RNC. (c) Nas redes 5G, o elemento principal é o gNB que se conecta entre si através da interface X_n, aos UEs através do NR-Uu e ao 5GC através das interfaces N_1 e N_2.



(a) 5G non-standalone.

Figura 2.3: Arquitetura *Non-Standalone* da redes 5G, na qual elementos da NG-RAN operam em conjunto com elementos da RAN e do núcleo da rede 4G.

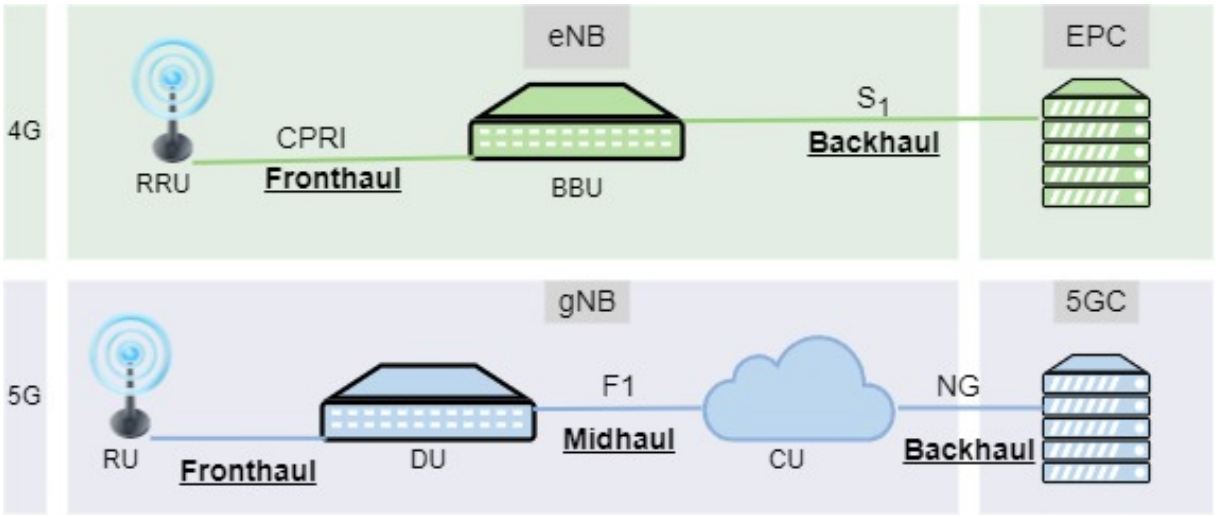


Figura 2.4: Comparação das Arquiteturas do eNB e gNB.

A Figura 2.6 resume as oito possibilidades de divisão, cada uma delas com suas vantagens e desvantagens. A divisão adotada pelo 3GPP foi a opção dois, na qual o gNB-CU é responsável pelos protocolos RRC e PDCP e o gNB-DU pelas camadas RLC, MAC e Camada Física (PHY).

A infraestrutura de uma rede 5G possui recursos heterogêneos, agrupados geograficamente em três níveis. O primeiro nível consiste no local de instalação dos nós de acesso de rádio. O segundo nível engloba uma área intermediária, chamada de *edge*, que possui capacidade computacional limitada. O terceiro consiste em uma área central com grande capacidade computacional que corresponde a um centro de dados. Para cumprir diversos requisitos, como latência e otimização de uso de *hardware*, a RAN pode ser dividida entre

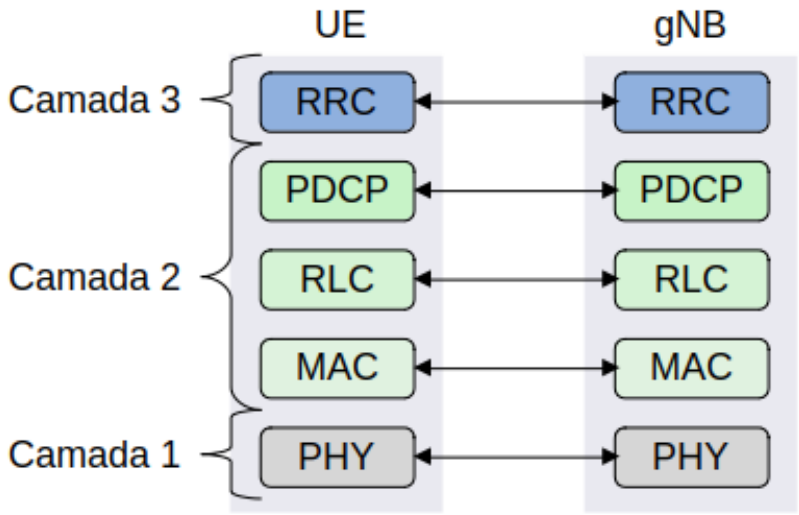


Figura 2.5: Pilha de protocolo do plano de controle da Rede de Acesso 5G.

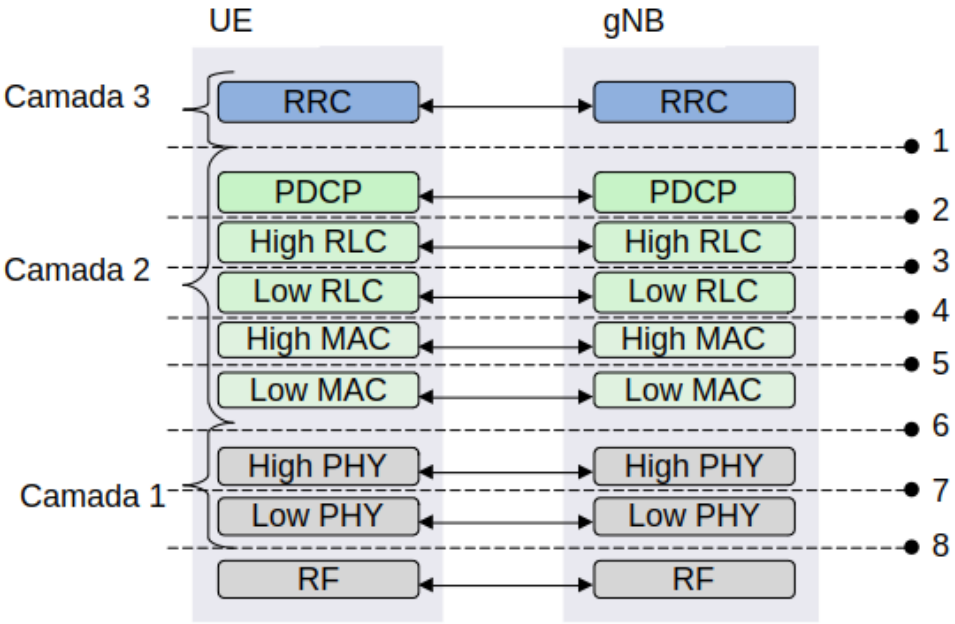


Figura 2.6: Opções de divisão funcional para a pilha de protocolo da Rede de Acesso 5G.

esses três níveis. As tecnologias C-RAN e O-RAN exploram essa descentralização da RAN e são detalhadas nas seções seguintes.

2.2.2 C-RAN

A tecnologia C-RAN permite que parte do processamento da rede de acesso seja realizada de forma centralizada em um centro de dados ou na nuvem, o que viabiliza a alocação

dinâmica dos recursos de processamento, otimizando o uso dos recursos da rede e, consequentemente, aumentando a eficiência energética e a economia em infraestrutura. A ideia principal do C-RAN é prover uma rede de acesso de rádio distribuída, porém com o processamento do sinal de rádio centralizado na nuvem [10]. Uma limitação do C-RAN é a necessidade de taxas de transmissão muito altas para o enlace *fronthaul*. Assim como na divisão entre gNB-CU e gNB-DU, a divisão dos protocolos executados na nuvem ou localmente impactam na capacidade exigidas pelo enlace. Quanto mais funções são executadas localmente, próximas ao usuário, menores são as exigências de taxa de bit da rede *fronthaul*. Por outro ângulo, quanto mais funções forem centralizadas em *data centers*, maior é a otimização do uso de *hardware* e mais flexível é a rede para balancear a carga de processamento [11]. Assim há um compromisso entre os benefícios do processamento em nuvem e a exigência de um enlace de altas taxas para atender os requisitos da comunicação.

Em uma rede 4G, o enlace *fronthaul* transporta amostras em fase e quadratura (I/Q) em banda base entre a RRU e a BBU usando a CRPI. A capacidade do *fronthaul* é proporcional a largura de banda disponível no eNB, a quantidade de antenas ativas, a resolução de quantização, a taxa de amostragem e carga da célula.

2.2.3 O-RAN

O modelo tradicional de RAN consiste em uma abordagem do tipo de caixa preta, na qual o *hardware* e *software* são *plug-and-play* com pouca, ou nenhuma, margem para personalização. A falta de controle total dos recursos e de parâmetros de comunicação impedem a adaptação da rede em tempo real de acordo com a demanda de serviços. Essa característica requer um projeto de rede baseado em valores máximos de utilização, o que pode ocasionar uma elevada taxa de ociosidade dos equipamentos e um aproveitamento sub-ótimo dos recursos.

Atualmente, há uma tendência crescente no emprego de sistemas abertos devido aos seus benefícios em relação a sistemas fechados e proprietários. O primeiro benefício é a colaboração de diversos atores para o desenvolvimento e aprimoramento de determinada tecnologia. O segundo é a transparência, uma vez que códigos e projetos abertos estão disponíveis para análise e podem ser avaliados em relação à privacidade e segurança. Em terceiro, pode-se elencar a diminuição da barreira de inovação que permite a entrada de diversos fabricantes no mercado. Nas novas gerações de redes móveis, há um grande esforço de pesquisa, tanto da indústria quanto das universidades, para a conversão da

RAN tradicional em uma arquitetura mais aberta e programável que permita o controle e gerenciamento unificado e agnóstico em relação aos fabricantes.

A O-RAN consiste na padronização de interfaces, principalmente da rede de acesso de rádio. A O-RAN *Alliance* é um grupo liderado por operadoras, cujos participantes também contam com assinantes e fabricantes, e tem por objetivo a implantações de redes abertas, interfaces padronizadas e possibilidade de uso de equipamentos de diferentes fornecedores [12]. O-RAN traz as vantagens dos sistemas abertos para a RAN dos sistemas de comunicações móveis que tradicionalmente são monolíticos e com poucas empresas atuando. Na RAN tradicional, um único vendedor é responsável pelo projeto do sistema, tanto do *software* quanto do *hardware* e das respectivas interfaces. Usando o paradigma de RAN aberta, o sistema é decomposto em blocos de funcionalidade e interfaces padronizadas. Também é proposto o desacoplamento entre *hardware* e *software*, permitindo a utilização de equipamentos e processadores COTS para a implementação da rede.

A proposta da O-RAN divide a RAN em três blocos: RU, DU e CU. A RU é responsável pelo *front end* digital, pelo *beamforming* e por parte da camada física. A RU é conectada à DU através do link *fronthaul*, a DU se conecta à CU através do *midhaul* e a CU se conecta ao núcleo da rede através do *backhaul*.

O 3GPP estimou a capacidade dos enlaces *fronthaul* de acordo com as opções de divisão funcional[13]. Nesse relatório, para as condições consideradas, a opção de divisão número oito, utilizada na divisão RRU e BBU requer um *fronthaul* com capacidade de 157.3 Gbps para *up* e *downlink* para uma rede 5G. A O-RAN Alliance adotou a divisão 7.2, pelo custo-benefício entre o custo do *fronthaul*, desempenho da rede, custo da RU e benefícios de virtualização. Conforme ilustrado na Figura 2.6, a divisão 7 secciona a pilha de protocolo na camada física (PHY), transformando essa camada em duas subcamadas: *High* PHY e *Low* PHY. Dessa forma, a RU é responsável pela *Low* PHY e pela camada de RF, enquanto a DU é responsável pelas camadas *High* PHY até a RLC, já o CU executa as camadas acima da RLC. Os equipamentos para implementação das CU e DU são servidores COTS com aceleradores de *hardware* para processamento em tempo real. Já as RUs podem ser dispositivos SDR ou outras placas comerciais que implementem a camada física inferior. As vantagens e oportunidades trazidas por tal paradigma são a interoperabilidade, a diminuição do consumo de energia, a criação de redes privadas e a redução dos custos de implementação da infraestrutura da rede.

2.2.4 Open Air Interface

Open Air Interface (OAI) é um conjunto de projetos de código aberto, controlado pela OpenAirInterface *Software Alliance*, que implementam e desenvolvem tecnologias para a RAN e o núcleo da rede celular [5]. Atualmente, existem os seguintes 4 projetos em andamento pelo grupo. O primeiro é o 5G RAN, cujo objetivo é implementar a pilha de protocolo 5G em suas diferentes configurações NSA e SA gNB, NSA e SA UE, divisão entre CU e DU. O segundo é o 5G *Core Network* que visa implementar o núcleo da rede 5G SA, orientado a serviços conforme as especificações do 3GPP, separando as funções do plano de controle e do plano de usuário. O terceiro é o Mosaic5G, cujo objetivo é desenvolver ferramentas de controle e orquestração para gerenciamento e programabilidade da infraestrutura da RAN e do CN. O quarto projeto visa realizar a integração e implementação contínua, implementando novos testes, mantendo e aprimorando a documentação e procedimentos.

2.2.5 OFDM

Um sistema de comunicação do tipo Multiplexação por Divisão de Frequência (FDM) divide a banda disponível no canal de comunicação em diversos subcanais, ou subportadoras, menores com um intervalo entre eles, denominado banda de guarda. Com essa técnica, é possível transmitir sinais independentes em cada uma dos subcanais de forma simultânea em um mesmo meio de comunicação. O OFDM é um método de transmissão similar à técnica tradicional de FDM, porém não necessita de bandas de guarda para a separação dos subcanais, uma vez que utiliza a sobreposição ortogonal das subportadoras, aumentando a eficiência espectral. Nesse tipo de sistema, o espaçamento entre as subportadoras pode ser projetado de maneira que cada subportadora seja alocada em um ponto do espectro onde as demais subportadoras sejam nulas. Assim, apesar da sobreposição, cada sinal pode ser recuperado através de um processamento adequado. Cada subportadora pode ter uma modulação ou codificação diferente e adaptada dinamicamente, conforme o estado do canal, o que torna a técnica robusta em relação a interferências inter-canal e inter-simbólica. Devido às suas diversas vantagens, o OFDM é utilizado em muitos sistemas de comunicação sem fio, como os padrões IEEE 802.11 e de redes móveis de 4G e 5G.

No domínio do tempo, o OFDM divide um fluxo de bits em N fluxos paralelos mul-

[5]Disponível em https://openairinterface.org/projects/.

tiplicados por senoides ortogonais, operando em múltiplas de uma frequência Δf_0. Os sinais de todas as portadoras somados formam o sinal a ser transmitido. O espaço entre as portadoras é $\Delta f_0 = \frac{k}{t_s}$, onde k é um número inteiro e t_s é o período de um símbolo. A largura de banda é $B = N\Delta f_0$. Como no OFDM um fluxo de alta taxa é substituído por N fluxos de taxa menor, então os símbolos são mais longos, o que permite a inserção de intervalos de guarda entre os símbolos e reduz a interferência inter-simbólica. Dessa forma, se a dispersão temporal do sinal ocasionada pelo multipercurso, ou seja, o intervalo de tempo entre a primeira recepção de um símbolo e o último eco, for menor do que o intervalo de guarda, a interferência intersimbólica é evitada. Um sistema OFDM requer um bom sincronismo de frequência, pois desvios de frequência implicam em perda da ortogonalidade entre as subportadoras, causando uma interferência interportadoras, Intercarrier Interference (ICI). Os desvios de frequência são normalmente ocasionados pelo descasamento dos osciladores dos transceptores e receptores. Algumas das subportadoras OFDM podem carregar símbolos piloto que permitem monitorar as condições do canal. Os símbolos piloto e o prefixo cíclico auxiliam na sincronização temporal e em frequência do sinal.

2.3 Rádio Definido por Software

Rádio Definido por Software é um sistema de radiocomunicação cujas funções na camada física são implementadas por *software*. A principal vantagem do uso do SDR é a facilidade de reprogramação de um rádio, o que possibilita a construção de diferentes tipos de transmissores utilizando um *hardware* genérico. Os SDRs utilizam computadores COTS para implementar funções de rádio e de processamento de sinais, como filtros, amplificadores, moduladores e transceptores. O uso de equipamentos genéricos e produzidos em larga escala diminui os custos de produção dos sistemas que utilizam essa tecnologia [14]. Além disso, a modificação e a atualização de aplicações baseadas em SDR são mais simples em virtude de as alterações serem apenas em *software* e, mesmo em casos de avarias de *hardware*, o reparo é facilitado devido à disponibilidade comercial dos dispositivos. Visto de outro ângulo, uma das grandes limitações da tecnologia SDR é causada justamente pelo uso de dispositivos genéricos de *hardware* que proporcionam a sua flexibilização. Dispositivos de *hardware* projetados com uma função específica são mais eficientes, em termos de processamento e utilização dos componentes, que dispositivos de *hardware* e processadores de uso geral devido à otimização da execução física das funções específicas.

Um dispositivo SDR idealmente realiza todo o processamento do sinal em *software*,

utilizando uma arquitetura de amostragem direta, na qual o sinal é digitalizado diretamente após a antena, sem nenhuma conversão de frequência [15]. O problema dessa abordagem está na necessidade de uma alta taxa de amostragem, o que implica conversores analógicos-digitais rápidos e eficientes. Outra desvantagem é a grande largura de banda necessária entre o dispositivo de rádio e o dispositivo de processamento do sinal. A banda de transmissão necessária cresce com a banda de frequência amostrada e com a taxa de amostragem usada.

2.3.1 GNU Radio

GNU Radio [6] consiste em um conjunto de ferramentas de desenvolvimento de *software*, livre e de código aberto, que implementa diversas funções de processamento de sinais. Utilizando o GNU Radio, é possível simular transmissores, receptores e analisadores de sinais, assim como avaliar implementações e técnicas de processamento de sinais. Além disso, quando combinado a um *hardware* programável, o GNU Radio permite criar receptores e transmissores definidos por *software*.

O *backend* do GNU Radio é escrito em C++ e sua interface gráfica, chamada de GNU *Radio Companion* (GRC), é escrita em Python. A conexão entre o *backend* e o GRC é realizada através da ferramenta *Simplified Wrapper and Interface Generator* (SWIG), que permite conectar programas escritos em C e C++ a linguagens de mais alto nível, como Python. O ambiente GRC possui diversos blocos que implementam funções de processamento, geração e visualização de sinais, manipulação de arquivos, entre outros. Através da conexão e configuração dos blocos, o GRC é capaz de criar programas em Python que implementam a aplicação desejada. GNU Radio utiliza um paradigma de programação orientada a objetos. Assim, os blocos são classes que implementam funções de processamento de sinais. As aplicações desenvolvidas usando o GNU Radio são chamadas de *Flowgraphs* e podem ser geradas pelo GRC ou escritas diretamente em Python ou C++.

A Figura 2.7 ilustra o *flowgraph* de implementação de um transmissor OFDM utilizando a ferramenta GRC. O primeiro bloco *File Source* referencia um arquivo a ser enviado e o transforma em um fluxo de caracteres. O próximo bloco adiciona uma referência de tamanho do pacote "`packet_len`", no fluxo de dados, indicando o início de cada pacote. Em seguida, é adicionada uma verificação cíclica de redundância, *Cyclic Redundancy Check* (CRC), ao final de cada pacote. O bloco *Packet Header Generator*

[6] Disponível em https://www.gnuradio.org/.

gera um cabeçalho pra cada pacote. O bloco *Repack Bits* reagrupa os bits em bytes de acordo com a quantidade de símbolos da modulação escolhida e o bloco *Chunks to Symbols* mapeia os bytes em símbolos. O próximo bloco, *Tagged Stream Mux*, multiplexa os cabeçalhos com a carga útil. Em seguida, o *OFDM Carrier Allocator* estabelece a quantidade de portadoras e a sua distribuição entre portadoras pilotos e portadora de dados. Além disso, esse bloco também é responsável por criar os símbolos OFDM. O bloco seguinte realiza a *Inverse Fast Fourier Transform* (IFFT) e então é adicionado um prefixo cíclico ao sinal. O bloco *Multiply Const* regula a amplitude do sinal e, por último, está o USRP *sink*, que estabelece o dispositivo de destino que realizará a transmissão e vários outros parâmetros de comunicação, tais como ganho, taxa de amostragem e largura de banda. O GRC utiliza o *flowgraph* para gerar um arquivo Python que implementa a aplicação projetada.

2.3.2 Dispositivo de Rádio Definido por *Software* USRP

Rádios definidos por *software* USRP compreendem uma gama de dispositivos que permitem uma rápida implementação de sistemas de comunicações, facilitando a criação de protótipos e a validação de projetos. Os USRPs são comercializados pela marca Ettus Research em diversos modelos, que variam em termos de arquitetura, capacidade de processamento e faixa de operação. A maioria dos modelos comercializados necessita de um computador hospedeiro que controle o *hardware* SDR e realize o processamento do sinal em banda básica. Para isso, é necessário que o computador tenha instalado o USRP *Hardware Driver* (UHD).

Em um dispositivo USRP, o processo de sintonia de frequência é realizado em duas etapas: a primeira é realizada pelo *front end* de RF, e tem por objetivos filtrar, amplificar e converter o sinal de RF. A segunda etapa, feita pelo *front end* digital, realiza a decimação no sinal digital recebido do Conversor Analógico-Digital (ADC) ou interpola o sinal que será convertido pelo Conversor Digital-Analógico (DAC). Na prática, o *front end* de RF é sintonizado o mais perto possível da frequência central escolhida e a *Field Programmable Gate Array* (FPGA) realiza o ajuste fino até a frequência desejada. Dessa forma, na cadeia de recepção, o *front end* analógico recebe e processa o sinal da antena e entrega o sinal ao ADC. Na cadeia de transmissão, é realizada a operação inversa, o sinal é recebido do DAC, convertido para banda passante e transmitido pela antena. O diagrama com os componentes do USRP podem ser visualizados na Figura 2.8.

Este trabalho utiliza dois dispositivos USRP, o B200 e N200. Ambos os dispositivos

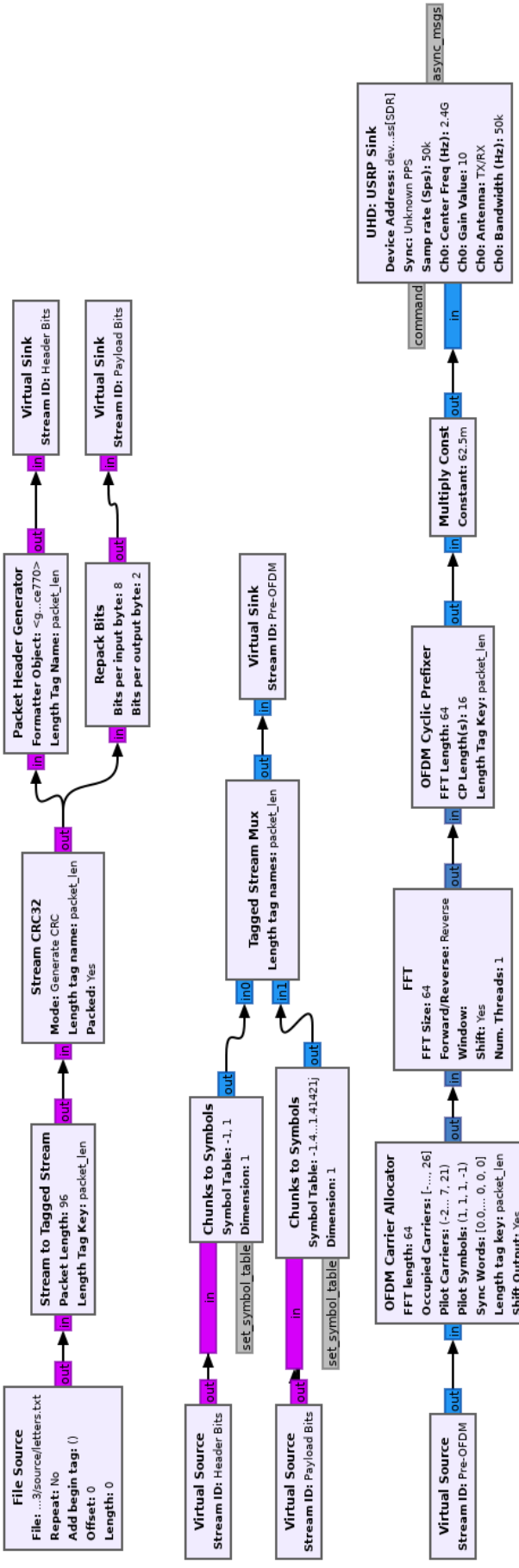


Figura 2.7: Implementação de um transmissor OFDM utilizando o GNURadio Companion.

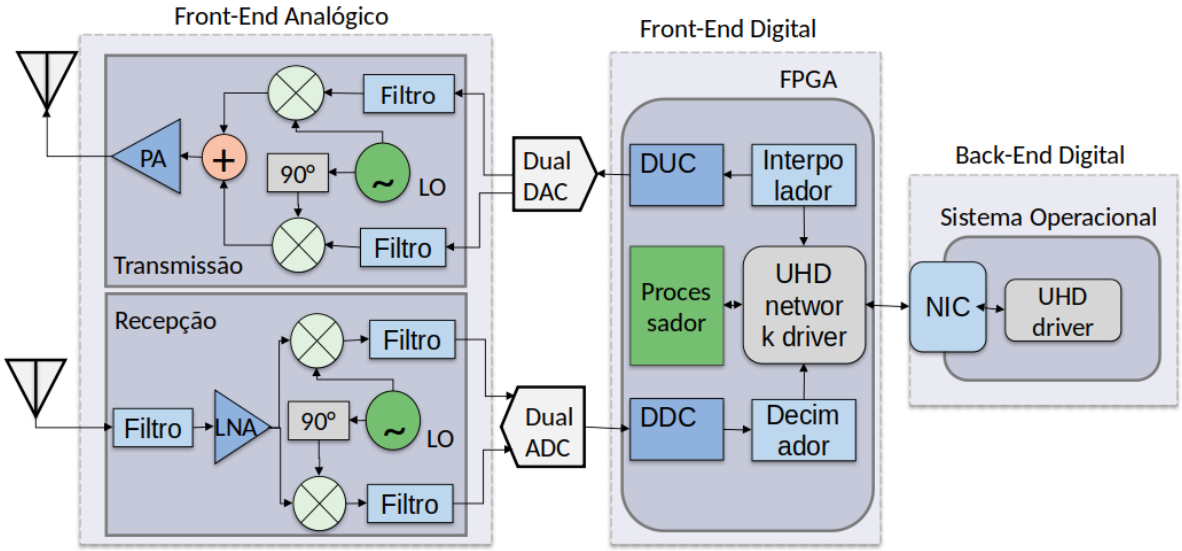


Figura 2.8: Arquitetura de um Rádio Definido por Software utilizando uma placa de rede Ethernet para comunicação com o computador hospedeiro.

precisam de um computador hospedeiro para controlar o SDR, porém se diferenciam em termos de arquitetura e capacidade. A arquitetura dos USRP N200 está ilustrada na Figura 2.8. Conforme pode ser observado, o N200 é composto de uma placa-mãe e uma placa-filha (*daughterboard*). A placa-mãe implementa o *front end* digital e a placa-filha constitui o *front end* de RF. A placa-mãe do dispositivo N200 opera em uma faixa de frequência de 0 a 6 GHz, contém uma FPGA capaz de processar 100 MHz de banda, um ADC dual de 100 MS/s de 14 bits e um DAC dual de 400 MS/s de 16 bits. Em relação às placas-filhas, existem diversos modelos disponíveis compatíveis com o N200, cada um com sua própria especificação de largura de banda e faixa de frequência. A placa-filha utilizada neste trabalho é a SBX, que opera na faixa de 400 MHz a 4,4 GHz com 40 MHz de largura de banda e possui dois *front ends*, um para transmissão ou recepção e outro somente para recepção. Os *front ends* possuem osciladores locais independentes que permitem uma operação full-duplex com frequências de transmissão e recepção distintas. A conexão do N200 com o computador é realizada através da placa de rede Ethernet, utilizando o protocolo da camada de aplicação VITA *Radio Protocol* (VRT) sobre datagramas *User Datagram Protocol* (UDP). Essa interface possui uma capacidade de amostragem de até 50 MS/s de transmissão e recepção operando no modo full-duplex. É recomendável realizar uma auto-calibração no dispositivo para minimizar o nível DC e o desbalanceamento de Fase e Quadratura (I/Q) nos sinais. O resultado da calibração é salvo em um arquivo no computador hospedeiro e o UHD automaticamente realiza as correções quando as aplicações SDR são executadas.

O B200 consiste em um USRP de placa única com um transceptor de conversão direta de chip único e processador de banda base digital, que transmite até 56 MHz de largura de banda de RF em tempo real. O dispositivo cobre frequências de 70 MHz a 6 GHz com uma conexão USB 3.0. O B200 possui um *front end* de RF para transmissão e outro para recepção com cadeias individualmente ajustáveis. As diferenças entre os dispositivos estão resumidas na Tabela 2.1.

Tabela 2.1: Especificação dos dispositivos USRPs

	B200	**N200**
Interface	USB 3.0	Gigabit Ethernet
Design	placa integrada	Placa-mãe + Placa-filha SBX
Faixa de frequência	70 MHz até 6 GHz	DC até 6 GHz SBX: 400 MHz to 4,4 GHz
Largura de banda	56 MHz	40 MHz
***Clock* principal**	5 até 30,72 MHz	100 MHz
Taxa de amostragem	61,44 MS/s	50 MS/s

A arquitetura USRP influencia diretamente a largura de banda máxima de uma aplicação SDR. Na cadeia de transmissão e recepção, a largura de banda varia em três estágios: no processamento do sinal no *front end* de RF, na capacidade de processamento da FPGA e na interface do computador com o USRP. A largura de banda máxima de uma aplicação SDR é a menor desses três estágios. Por exemplo, no caso do N200, têm-se as seguintes características: a placa-filha SBX possui largura de banda de 40 MHz; o processamento digital realizado pela FPGA recebe amostras a uma taxa de 100 MS/s do ADC e de 400 MS/s do DAC; e a conexão Gigabit Ethernet com o computador hospedeiro possui uma taxa máxima teórica de 25 MS/s, utilizando amostras em I/Q com 16 bits em modo *full-duplex*. Assim, a largura de banda das aplicações desenvolvidas com essa combinação de dispositivos está limitada à largura de banda da conexão Ethernet entre o USRP N200 e o computador hospedeiro.

Outro ponto importante a ser mencionado consiste nas restrições sobre os valores das frequências de amostragem que podem ser utilizados nos dispositivos USRP. Segundo o manual do fabricante [16], é necessário que a razão entre a frequência do *clock* principal e a frequência de amostragem seja um número inteiro, preferencialmente par. Além disso, valores de interpolação ou de decimação maiores que 128 devem ser divisíveis por 2. Caso a taxa de interpolação ou de decimação seja maior que 256, esse valor deverá ser divisível por 4. Considerando que o modelo N200 possui uma frequência de *clock* fixa de 100 MHz, as frequências de amostragem possíveis são limitadas pelas restrições mencionadas.

2.4 Trabalhos Relacionados

Os trabalhos relacionados, assim como a dissertação, são apresentados em duas partes. A primeira, referente à utilização de dispositivos SDR em ambientes virtualizados, é apresentada nesta seção, enquanto os trabalhos relacionados à descoberta de dispositivos IoT são abordados na seção 4.3. Os trabalhos de referência para esta primeira parte são aqueles que avaliam o desempenho de dispositivos SDRs, realizam testes de desempenho de ferramentas de virtualização ou que implementam sistemas de comunicação móvel em SDR.

Schimid *et al.* estudam o impacto da latência em dispositivos SDR na camada de enlace em dois cenários, o primeiro deles consiste em um esquema FSK simples e o segundo é baseado no padrão IEEE 802.15.4 [17]. Os autores avaliam o tempo necessário para gerar uma amostra no computador e enviá-la através do USRP e, de forma simétrica, o tempo em que a amostra é recebida pelo USRP até o tempo em que se torna disponível para a camada de enlace no computador. Os testes são realizados com o auxílio de um osciloscópio externo e uma porta paralela do computador. Segundo os autores, a latência final de recepção pode ser calculada como o somatório da latência inserida pelo USRP, pela comunicação USB e pelo *software* GNU Radio. A latência de transmissão é diretamente afetada pelo *buffer* de 32 kB existente entre o GNU Radio e a interface USB, de forma que as amostras são transmitidas quando há um número suficiente para preencher um pacote USB. Os resultados obtidos demostram que as implementações em SDR do protocolo IEEE 802.15.4 possuem uma latência média de 25 ms, que é considerada uma latência alta, uma vez que um rádio convencional possui uma latência de 8 ms. Essa alta latência observada inviabiliza a implementação de protocolos com requisitos restritos de tempo.

Becker *et al.* propõem uma arquitetura de teste para *benchmarking* de desempenho de dispositivo sem fio baseado em rádio definido por software. O *testbed* simplifica os testes de desempenho da camada física, se valendo da facilidade de controle de parâmetros do SDR, o que proporciona consistência e reprodutibilidade para os testes. Para validar a proposta, os autores comparam o desempenho de placas Wi-Fi de quatro fabricantes diferentes em múltiplos testes e conseguem identificar diferenças significativas, incluindo diferenças notáveis de sensibilidades dos receptores. Através dos resultados, é possível verificar que os fabricantes implementam de maneira diferente recursos da camada física que não são definidos com precisão no padrão, o que deve ser cuidadosamente considerado ao desenvolver modelos analíticos e de simulação de redes Wi-Fi [18].

Chang *et al.* comparam o desempenho de uma implementação do núcleo da rede 5G, utilizando Open5GCore, sobre KVM e Docker. O trabalho verifica o comportamento das ferramentas de virtualização com o tráfego gerado pelo Open5GMTC, uma ferramenta de prototipagem para IoT. Os resultados obtidos mostram que que os núcleos da rede virtualizados no KVM e no Docker podem funcionar tão bem quanto em uma máquina física, desde que o número de dispositivos conectados seja menor que setenta. Em termos de CPU, KVM e Docker apresentam um consumo um pouco superior que o sistema nativo. Em termos de memória, a virtualização também apresenta um uso mais elevado devido à necessidade de manutenção de máquinas virtuais e containeres, sendo o KVM o ambiente com maior uso de memória, que pode ser atribuído à necessidade de virtualização de todo o sistema, enquanto o Docker virtualiza apenas no nível da aplicação. A conclusão do trabalho mostra que a virtualização do núcleo da rede, especialmente do Docker, é viável devido à sua baixa sobrecarga de virtualização [19].

Mattos *et al.* avaliam ferramentas difundidas de virtualização para a implementação de roteadores virtuais em relação ao uso de CPU, memória, rede e disco [20]. Os autores observaram que a ferramenta OpenVZ, baseada em virtualização do espaço de usuário, introduz menor sobrecarga de CPU, disco e memória, porém apresenta uma degradação no desempenho de rede. Por outro lado, a plataforma Xen obteve o melhor resultado na análise de rede, o que é fundamental na utilização de redes virtuais. Os autores evidenciam que os ambientes de virtualização devem ser avaliados de acordo com as aplicações e serviços que serão executados e suas especificidades.

Eiras *et al.* realizam uma análise de desempenho de dois ambientes de virtualização, KVM e Docker, em um cenário de proxy HTTP que consiste em um serviço sensível à sobrecarga de CPU e de rede. O trabalho avalia o tempo total necessário para realizar solicitações HTTP e receber as respostas no proxy, utilizando ambas as tecnologias de virtualização. Os experimentos mostraram que um contêiner Docker pode ser duas vezes mais rápido que uma máquina virtual baseada em KVM, pois o Docker compartilha todos os recursos com o sistema operacional hospedeiro e outros elementos, como tabela de IP e tabela de roteamento. Dessa forma, os autores concluem que o Docker é mais eficiente para funcionar como proxy em um ambiente NFV quando comparado a máquinas virtuais baseadas em KVM [21].

Morabito *et al.* comparam ferramentas de virtualização em termos de processamento, armazenamento, memória e rede, utilizando o desempenho de um Linux nativo como referência. O trabalho utiliza as seguintes ferramentas: Linux no KVM, como um exemplo

de sistema baseado em hipervisor, Docker e LXC, como soluções baseadas em contêineres, e OSv, como um sistema híbrido baseado em nuvem. Os resultados mostram que os contêineres alcançam desempenho geralmente melhor quando comparados com as demais soluções, embora a versatilidade e facilidade de gerenciamento sejam pagas em termos de segurança [22].

Bachiega *et al.* realizaram um mapeamento sistemático nas principais bases de dados da literatura para identificar as técnicas de avaliação de desempenho e as cargas de trabalho mais utilizadas. As principais técnicas de avaliação de desempenho são modelagem analítica, simulação e medição. A técnica de medição é extensivamente estudada e com o uso de benchmarks para CPU, memória, banco de dados, redes, aplicações científicas e outras ferramentas de virtualização [23].

Bloessl *et al.* implementam o protocolo IEEE 802.11p, que consiste em um padrão de comunicação sem fio entre veículos, utilizando a ferramenta GNU Radio e um dispositivo USRP N210 [24]. A aplicação desenvolvida suporta modulações BPSK, QPSK, 16-QAM e 64-QAM definidas no padrão, porém não cumpre os requisitos de tempo necessários para a troca de mensagem de alocação de canal RTS/CTS e para as mensagens ACK que sinalizam o recebimento de quadro. Dessa forma, a implementação apresentada se restringe a comunicações de difusão, não sendo capaz de realizar a associação a um ponto de acesso ou comunicação *unicast* com outras estações.

Dzogovic *et al.* implementam uma rede 5G utilizando a ferramenta OpenAirInterface e dispositivos USRP com ambientes virtuais de contêineres e máquinas virtuais, porém os autores não fornecem um comparativo de desempenho e consumo de recursos computacionais do sistema em execução e também não quantificam o impacto da virtualização no desempenho do sistema [25]. Lai *et al.* propõem a virtualização do núcleo da rede móvel utilizando contêineres e o software OpenAirInterface (OAI), assim como um algoritmo para otimizar a utilização de CPU. Os autores apresentam uma análise do uso de CPU, mas somente do computador contendo os contêineres que executam funções do núcleo da rede. Não há uma análise sobre o desempenho e uso de recursos da RAN implementada [26].

Capítulo 3

SDR em ambientes virtualizados

As tecnologias de virtualização existentes, como máquinas virtuais e contêiner, são ferramentas amplamente empregadas para diminuir a ociosidade de equipamentos, permitir o compartilhamento de recursos e desacoplar as aplicações dos hardware em que são executados. Devido à rigidez das RANs tradicionais, as redes são projetadas para atender o pico de demanda, levando a uma ociosidade dos equipamentos no restante do tempo. Atualmente, já existem projetos para a implementação de RAN utilizando SDR, como descrito no Capítulo 2, porém não é previsto o uso do SDR em ambientes virtualizados nem em uma abordagem de NFV, como já é implementado o núcleo da rede.

Devido às inúmeras aplicações e benefícios que a virtualização de rádio pode fornecer, é essencial observar o impacto da virtualização e da conteinerização na comunicação de aplicações SDR. O conhecimento da largura de banda que o conjunto formado pelo computador hospedeiro, ambiente de virtualização e USRP suporta é fundamental para o projeto de aplicações SDR. A Seção 2.3.2 expõe que a largura de banda de uma aplicação desenvolvida com o N200 é limitada pela conexão Ethernet entre o computador hospedeiro e o dispositivo USRP e o seu limite teórico. Porém, o conhecimento da taxa prática real alcançada na comunicação Ethernet é de extrema importância para definir as aplicações em SDR que são realizáveis com a largura de banda experimental disponível.

A arquitetura do SDR e seu modo de comunicação com o computador hospedeiro influenciam a largura de banda máxima dos aplicativos SDR, uma vez que todos os sinais de controle de (e para) o aplicativo atravessam essa mesma interface. Dessa forma, a utilização de taxas de amostragem incompatíveis com a capacidade do *hardware* pode levar à perda de pacotes e a erros de operação. Durante a recepção, caso a aplicação não consiga processar os dados na velocidade de recepção das amostras, ocorre um erro de *overflow* e o descarte das amostras. Além disso, quando a aplicação não consegue

produzir amostras suficientes para suprir a taxa de transmissão configurada, ocorre um *underflow*.

3.1 Metodologia de avaliação de desempenho

O presente trabalho propõe uma avaliação do uso de dispositivos SDR em ambientes virtualizados com o objetivo de mensurar o impacto da virtualização na comunicação dos rádios e verificar a viabilidade dessa abordagem em redes de acesso via rádio na nuvem. A análise da comunicação de rádios SDR em ambientes virtualizados é realizada através de diversos testes, variando modelos de SDR, modos e parâmetros de comunicação. Todos os testes são realizados utilizando duas abordagens de virtualização, VMs e contêineres, assim como dois modelos de SDR, N200 e B200.

Nos testes realizados, as aplicações de rádio são executadas em um sistema operacional nativo, em uma máquina virtual (VirtualBox) e em um contêiner (Docker). Para cada um desses ambientes, a forma como as amostras são entregues à aplicação varia de acordo com a tecnologia de virtualização. No caso do dispositivo N200, a ligação ao computador é feita através de uma conexão Ethernet e, assim, através dessa interface, as amostras são encaminhadas à aplicação de rádio. A Figura 3.1 ilustra a conexão do Linux Nativo, que consiste no primeiro ambiente de teste, com o USRP N200. É possível notar que o drive UHD recebe as amostras oriundas da interface de rede `eth0` e direciona para a aplicação. Essa configuração é a mais básica dos cenários de teste com o N200 e é utilizada como referência para avaliar o resultado desse dispositivo nos ambientes virtualizados.

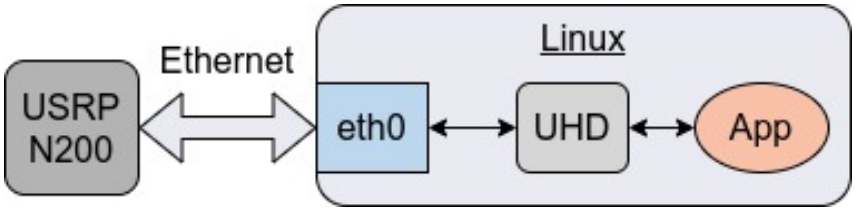


Figura 3.1: Configuração do Linux com o USRP N200. A interface de rede `eth0` do Linux é configurada na mesma sub-rede IP do USRP N200.

O segundo ambiente de teste utiliza máquinas virtuais e, nessa configuração, é necessária uma ponte (*bridge*) para interligar a interface `eth0`, gerenciada pelo OS nativo, à interface `eth0` da VM para permitir a comunicação entre a aplicação e o USRP. A Figura 3.2 ilustra a conexão da máquina virtual com o USRP N200. O endereço IP da interface `eth0` da VM é configurado para pertencer à mesma sub-rede do N200.

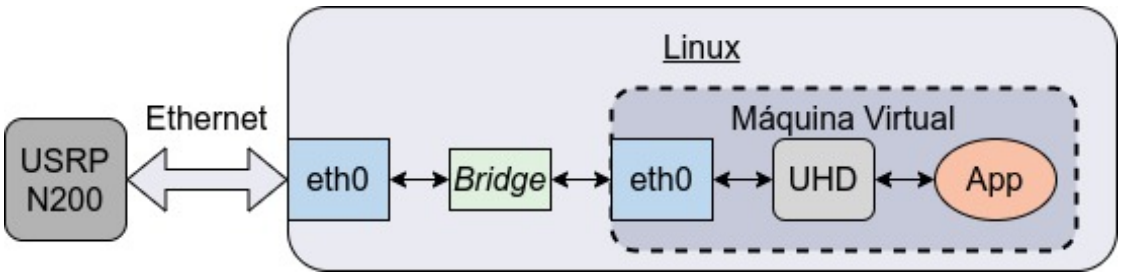


Figura 3.2: Configuração da máquina virtual com o USRP N200. A interface de rede `eth0` do máquina virtual é configurada na mesma sub-rede IP do USRP N200.

O terceiro e último ambiente testado utiliza contêineres. Sua rede é configurada usando uma ponte Linux e um par de interfaces virtuais Ethernet (`veth0` e `veth1`), como ilustrado na Figura 3.3. A ponte é responsável por encaminhar pacotes entre a interface de rede `eth0`, que corresponde à placa de rede Ethernet, e `veth0`, que atua como um túnel Ethernet para a interface `veth1` dentro do contêiner.

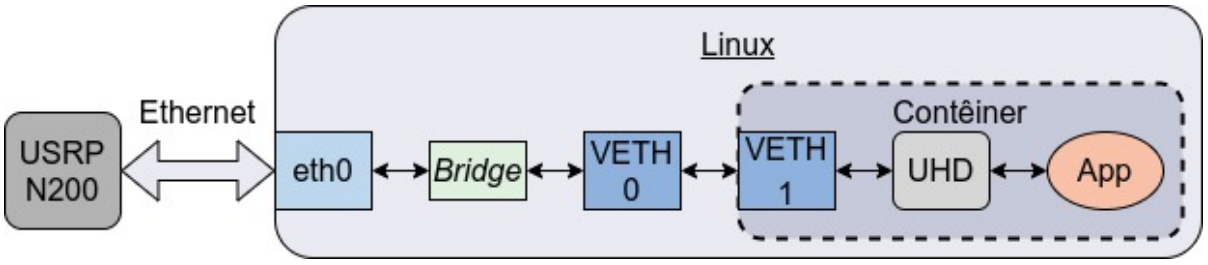


Figura 3.3: Configuração de rede do contêiner. A *bridge* é responsável por encaminhar os pacotes entre as interfaces de rede `eth0`, que corresponde à placa de rede Ethernet, e `veth0`, que atua como um túnel Ethernet para a interface `veth1` do contêiner.

Em relação ao USRP B200, a sua conexão ao computador é realizada através de USB. Para comunicação USB na VM, é necessário instalar o pacote de extensão VirtualBox, habilitar o controlador USB 3.0 e criar um filtro de dispositivo USB usando a identificação do fabricante (*Vendor* ID) e do produto (*Product* ID). Nos contêineres, o acesso aos dispositivos USB é realizado adicionando a *flag* "–device" ao iniciar o contêiner ou adicionando o dispositivo no documento de criação do contêiner "`docker-compose.yml`".

Em relação aos testes realizados, primeiramente, é avaliado como a virtualização afeta a conexão Ethernet e o consumo de CPU. Em seguida, são testados os limites inerentes ao USRP e de sua conexão com o computador. Para isso, são utilizados três modos de comunicação (transmissão simplex, recepção simplex e *full-duplex*) com o objetivo de medir a taxa de amostragem máxima alcançada pelo dispositivo SDR e a sobrecarga de CPU em cada cenário. Mede-se o número de amostras recebidas e transmitidas e seu uso de CPU correspondente. Por último, implementa-se uma comunicação OFDM entre

dispositivos SDR utilizando uma configuração de *loopback*, variando parâmetros de comunicações, como tamanho do pacote, número de portadoras e ambiente de virtualização. Então, é avaliada a perda de dados em uma transferência de arquivos para cada um dos cenários testados. Através dos resultados obtidos, identifica-se como cada tecnologia de virtualização influencia no desempenho do sistema ao compará-los com o desempenho do mesmo sistema em um sistema operacional nativo.

3.2 Resultados Experimentais

Os computadores utilizados nos testes são equipados com processador I7-4770 CPU @ 3,40 GHz, 16 GB de memória, placa de rede equivalente a Realtek RTL8111/8168/8411. As tecnologias de virtualização adotadas são VirtualBox 6.1 e Docker 20.10. Ambos os sistemas operacionais virtualizados e nativos são Ubuntu Linux 20.04. Todos os ambientes têm endereços IP estáticos pertencentes à mesma sub-rede do dispositivo USRP. As máquinas virtuais e os contêineres são configurados com 4 GB de memória e 4 núcleos de vCPU.

3.2.1 Comunicação Ethernet em ambientes virtualizados

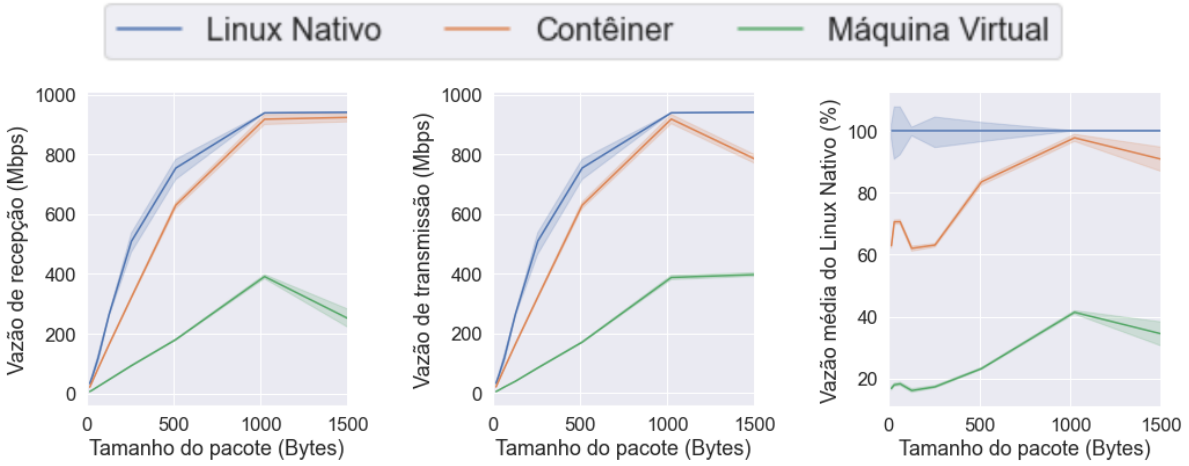
A vazão de uma rede é altamente dependente do padrão de tráfego e das configurações da rede [27]. Assim, a avaliação do impacto que as ferramentas de virtualização exercem na comunicação Ethernet é o primeiro fator estudado neste trabalho, pois essa característica influencia como o dispositivo SDR N200 comunica através dessa interface. Dessa forma, o primeiro teste realizado consiste em testar a vazão da comunicação Ethernet para diferentes tamanhos de pacotes nos três ambientes de teste, sem o uso de dispositivos SDR. Além disso, também é medido o uso de CPU em cada um dos cenários para verificar o impacto que os tamanhos dos pacotes causam no uso de processamento do *hardware*. O objetivo do teste consiste em verificar como a virtualização e a alocação de recursos de *hardware* impactam a capacidade de comunicação via Ethernet de um sistema virtualizado. Os testes são realizados utilizando o *software* `iPerf` [1] que possui uma arquitetura cliente/servidor e permite medir a máxima largura de banda alcançável em redes IP. O servidor iPerf é executado em uma máquina com configurações idênticas às do computador hospedeiro que executa o cliente iPerf e ambas as máquinas pertencem à mesma rede local, conectadas através de um comutador Gigabit Ethernet.

[1] Disponível em https://iperf.fr/.

Como a comunicação entre o dispositivo SDR e computador ocorre através do protocolo UDP na camada de transporte, o teste é realizado utilizando datagramas UDP. Cada ambiente é testado com datagramas variando de 16 a 1500 Bytes, o que possibilita verificar como o tamanho dos pacotes afeta a vazão alcançada pelos sistemas nos dois sentidos de comunicação *full-duplex*. Em um ambiente SDR em que a conexão com o computador hospedeiro é feita através de uma conexão Ethernet, o uso de CPU para processamento dos pacotes de rede é crítico, pois todas as informações recebidas e transmitidas pelo rádio utilizam essa interface e uma grande sobrecarga pode limitar a capacidade de processamento do sinal em banda base pelo hospedeiro.

Os experimentos foram executados dez vezes para cada combinação ambiente e dispositivo USRP. Os parâmetros utilizados na ferramenta iPerf foram largura de banda de um Gb/s, protocolo UDP na camada de transporte, duração de trinta segundos para cada teste, tamanho dos datagramas variando de 16 a 1500 Bytes e configuração bidirecional, na qual o cliente iPerf atua como transmissor e receptor alternadamente. O resultado do teste consiste nos valores médios de vazão medidos pelo receptor, apresentados dentro de um intervalo de confiança de 95%.

A Figura 3.4 mostra a vazão alcançada utilizando diferentes tamanhos de pacotes nos três ambientes: Linux nativo, máquina virtual e contêiner. É possível observar que, quanto menores forem os pacotes, menor é a vazão alcançada para todos os ambientes, tanto para a recepção, visto na Figura 3.4 (a), quanto para a transmissão, ilustrado na Figura 3.4 (b). Ademais, a máquina virtual atinge uma vazão para recepção e transmissão de pacotes consideravelmente menores que nos outros dois ambientes analisados. A Figura 3.4 (c) mostra a taxa de transmissão dos ambientes virtualizados em relação ao valor médio da vazão do Linux nativo. Através desse resultado, é possível observar que a vazão do contêiner representa mais de 80% da vazão média do Linux nativo para pacotes acima de 500 bytes, enquanto a vazão da máquina virtual atinge, no máximo, 40% da vazão alcançada com o Linux nativo. Assim, é possível observar que a máquina virtual tem um decréscimo considerável em relação à vazão alcançada nativamente, o que pode provocar queda no desempenho de aplicações intensivas em rede executadas nesse ambiente. Esse resultado é reflexo das técnicas usadas para a virtualização das interfaces de rede. Na abordagem baseada em máquinas virtuais, a interface de rede virtual é paravirtualizada [27], criando um novo *driver* no contexto da máquina virtual que recebe os pacotes do sistema físico através de cópia de memória executada pelo hipervisor. Na abordagem de contêiner, apesar dos resultados próximos ao do Linux nativo, o par de interfaces virtuais utilizado para a comunicação dos contêineres gera uma sobrecarga que



(a) Vazão de recepção. (b) Vazão de transmissão (c) Vazão relativa

Figura 3.4: Análise do impacto do tamanho dos pacotes no desempenho do sistema: (a) vazão de recepção alcançada para diferentes tamanhos de pacotes UDP, (b) vazão de transmissão alcançada para diferentes tamanhos de pacotes UDP, (c) comparação da vazão de transmissão dos ambientes em relação à vazão média do Linux Nativo.

reduz a vazão alcançada pelo sistema.

A Figura 3.5 mostra o uso de CPU durante a realização dos testes, medido a partir do sistema operacional hospedeiro. É possível observar que os consumos observados nos testes com contêiner e Linux nativo são muito próximos, porém o uso de CPU é três vezes maior em um ambiente com máquina virtual em função da execução de dois sistemas operacionais distintos (hospedeiro e visitante) e da paravirtualização da interface de rede.

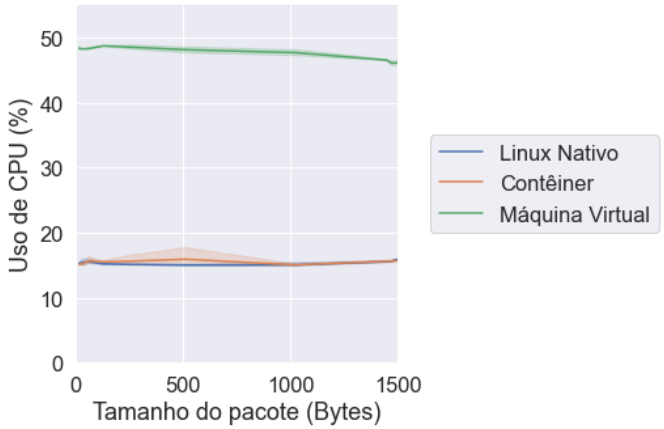


Figura 3.5: Uso de CPU da máquina física durante os testes de rede de acordo com os tamanhos dos pacotes.

Através desse teste inicial, é possível concluir que, independentemente da aplicação executada, a máquina virtual apresenta uma inerente sobrecarga de processamento e

uma redução na vazão de rede em comparação ao desempenho obtido em um sistema operacional nativo.

3.2.2 Comunicação do USRP em ambientes virtualizados

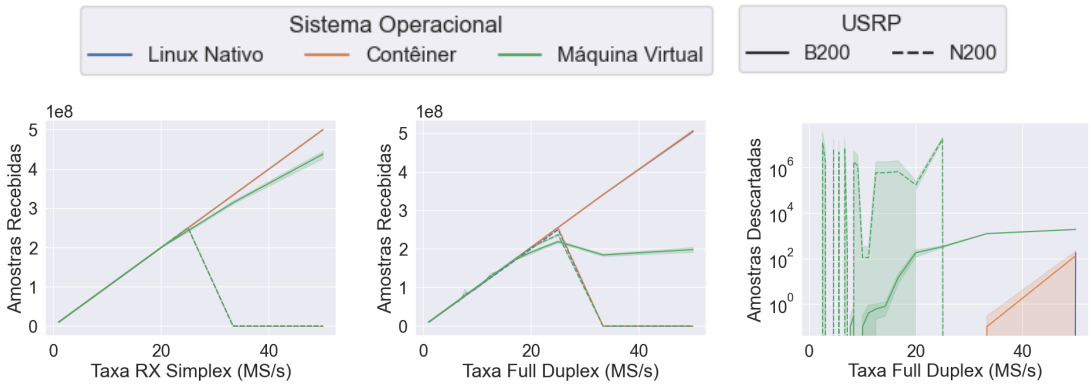
A avaliação de comunicação do USRP visa estabelecer um limite experimental de largura de banda entre o computador e os dispositivos USRP. O segundo objetivo é verificar o impacto da virtualização na comunicação entre o computador e os dispositivos SDR. Os testes são realizados através da ferramenta de *benchmark* fornecida com o UHD, usando diferentes taxas de amostragem e modos de comunicação. Como as taxas de amostragem no USRP são limitadas àquelas cujos valores são múltiplos da frequência de *clock* do dispositivo e o dispositivo N200 possui uma frequência de *clock* fixa de 100 MHz, escolhe-se 25 taxas de 1 MS/s a 50 MS/s, que consiste no limite máximo especificado na documentação do N200 para amostras de 8 bits.

Os testes contemplam doze cenários diferentes, usando três ambientes diferentes: Linux nativo, máquina virtual e contêiner; três modos de comunicação: transmissão simplex, recepção simplex e *full-duplex*; e dois dispositivos USRP: N200 e B200. Os testes foram realizados com amostras I/Q de 16 bits. Os experimentos foram executados dez vezes para cada combinação de modo de comunicação, ambiente e dispositivo USRP. Os resultados são os valores médios, apresentados dentro de um intervalo de confiança de 95%.

A Figura 3.6 apresenta os resultados do teste de recepção. A Figura 3.6(a) mostra a quantidade de amostras recebidas no modo simplex, a Figura 3.6(b) apresenta as amostras recebidas no modo *full-duplex* e a Figura 3.6(c) mostra o número de amostras descartadas no modo *full-duplex*.

Em virtude dos resultados obtidos terem sido muito próximos, na Figura 3.6(a) há uma sobreposição das curvas dos testes realizados nos três ambientes e nos dois USRPs. O B200 teve um comportamento linear quando utilizado com o Linux nativo e com o contêiner, enquanto o N200 apresentou um decaimento a partir de 25 MS/s em todos os ambientes. Esses resultados são consistente com a documentação do dispositivo [2] que estabelece o limite teórico do dispositivo para 50 MS/s para amostras de 8 bits e 25 MS/s para amostras de 16 bits. O único cenário de teste que obteve um comportamento diferente devido ao ambiente foi o B200 com a máquina virtual, o que pode ser justificado pela sobrecarga de CPU, visto que o decaimento da quantidade de amostras recebidas

[2]Disponível em https://files.ettus.com/manual/



(a) Simplex. (b) *Full-duplex*. (c) *Full-duplex*.

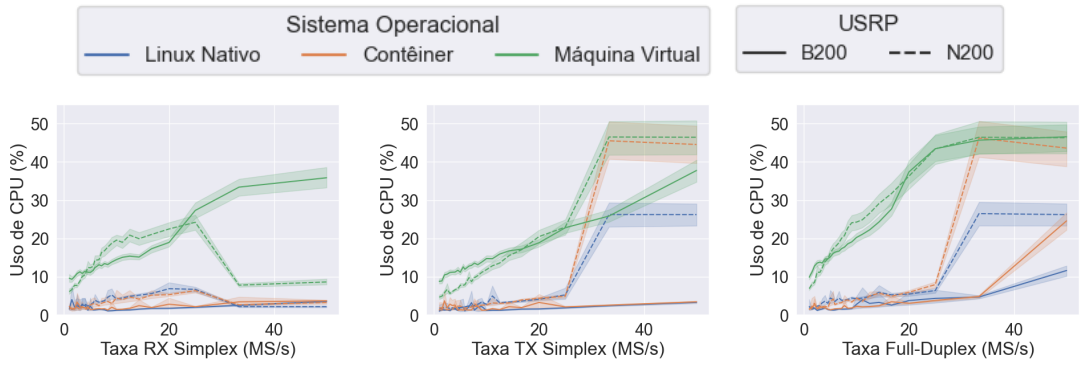
Figura 3.6: Análise da recepção em relação à taxa de amostragem: (a) quantidade de amostras recebidas no modo simplex, (b) quantidade de amostras recebidas no modo *full-duplex*, e (c) quantidade de amostras descartadas no modo *full-duplex*. Há uma sobreposição dos resultados em (a) e (b), pois o desempenho observado nos cenários foi equivalente, exceto para o USRP B200 em máquina virtual.

coincide com um aumento significativo no uso de CPU que atinge 30% da sua capacidade, conforme Figura 3.7(a).

De maneira semelhante, a Figura 3.6(b) mostra os resultados *full-duplex*, onde é possível observar que são semelhantes aos testes em modo simplex, exceto para os cenários com VM, nos quais as perdas de pacotes iniciam a taxas de 16 MS/s. A pior degradação no desempenho ocorre para o USRP B200, cujo número de amostras recebidas cai consideravelmente para taxas superiores a 25 MS/s, em comparação com o número de amostras recebidas no modo simplex. Observando o uso de CPU desse teste, 3.7(c), é possível verificar que a degradação ocorre para valores de 30% de CPU, assim como no teste anterior, indicando um limiar de processamento a partir do qual ocorre uma degradação na comunicação.

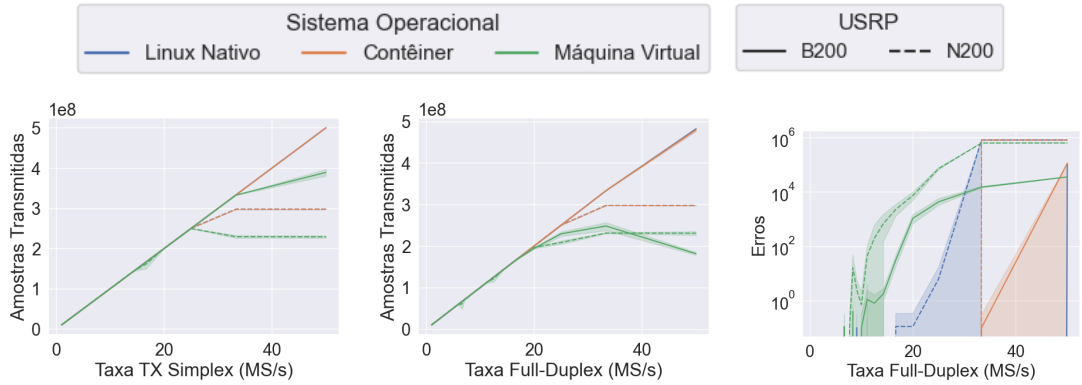
A Figura 3.6(c) mostra o número de amostras descartadas pelo receptor no modo *full-duplex*, em que a máquina virtual apresenta a quantidade mais significativa de amostras descartadas. O descarte de amostras acontece quando o sistema não é capaz de processar os dados na velocidades em que são recebidos. Assim, pode-se concluir que a VM gera uma sobrecarga de CPU no sistema que diminui o número de amostras recebidas na comunicação *full-duplex* e degrada significativamente o desempenho dos aplicativos SDR nesse ambiente. Por outro lado, o cenário de teste utilizando contêiner obteve resultados próximos ao cenário baseado no Linux nativo.

A Figura 3.8(a) mostra o número de amostras transmitidas em uma transmissão



(a) Recepção - Simplex. (b) Transmissão - Simplex. (c) *Full-duplex*.

Figura 3.7: Uso de CPU de acordo com o modo de comunicação.



(a) Simplex. (b) *Full-duplex*. (c) *Full-duplex*.

Figura 3.8: Análise da transmissão em relação à taxa de amostragem: (a) quantidade de amostras transmitidas no modo simplex, (b) quantidade de amostras transmitidas no modo *full-duplex*, e (c) quantidade de *underruns* que ocorreram no modo *full-duplex*. Há uma sobreposição dos resultados em (a) e (b), pois o desempenho observado nos ambientes Linux nativo e contêiner foi equivalente para todas as taxas testadas.

simplex. É possível observar que todos os três ambientes têm resultados semelhantes, e apresentados no gráfico sobrepostos, para taxas de amostragem de até 25 MS/s. Para taxas acima desse valor, a quantidade de amostras transmitidas cai para o dispositivo N200 para todos os ambientes, porém de forma mais acentuada para a máquina virtual. Para o B200, o cenário com a VM mostra uma queda no desempenho para taxas superiores a 33 MS/s, assim como no teste anterior, ao verificar o uso de CPU na Figura 3.7(b), observa-se a correlação da degradação da comunicação com o uso de CPU próximo a 30%. A Figura 3.8(b) mostra o número de amostras transmitidas em uma comunicação *full-duplex*. A principal diferença observada entre a transmissão *full-duplex* e a transmissão simplex ocorre quando a aplicação é executada na VM com o dispositivo B200, apresentando

uma diminuição considerável no número de amostras transmitidas correlacionada com o consumo de CPU apresentado na Figura 3.7(c). O dispositivo N200 também apresenta um desempenho inferior no ambiente de VM.

A Figura 3.8(c) mostra o número de erros de *underrun* no modo *full duplex*, que indicam que o computador não gera amostras suficientes para atingir a taxa de amostragem definida para o teste. Mais uma vez, a VM apresentou mais erros de *underrun* para ambos os dispositivos USRP e apresentou o pior desempenho dos três ambientes de testes. Igualmente, é possível notar que, mesmo no ambiente do Linux nativo, o dispositivo N200 apresentou erros a partir de 16 MS/s, o que sinaliza limitações da própria arquitetura do dispositivo.

A Figura 3.7 mostra o uso da CPU durante a execução dos testes. A Figura 3.7(a) mostra que, na recepção simplex, a VM tem um uso de CPU maior para ambos os dispositivos USRP. Além disso, os cenários com Linux nativo e com contêiner obtiveram resultados semelhantes no modo de recepção simplex. A Figura 3.7(b) revela que a transmissão simplex é mais custosa computacionalmente que a recepção simplex para quase todos os cenários testados, sobretudo para o dispositivo N200 para taxas de amostragem superiores a 25 MS/s. A Figura 3.7(c) mostra o consumo de CPU em um cenário *full-duplex*, onde se nota um aumento geral no uso de CPU para todos os cenários de teste. No entanto, o dispositivo B200 dentro do ambiente VM apresentou o maior aumento no uso de CPU, chegando a dobrar, quando comparado ao desempenho dos modos simplex para as mesmas taxas de amostragem, o que pode ser justificado pela forma como a conexão USB é implementada na máquina virtual. É possível observar também que em todos os testes com o N200, quando o dispositivo alcança o limite de taxa de amostragem, o uso de CPU se torna constante.

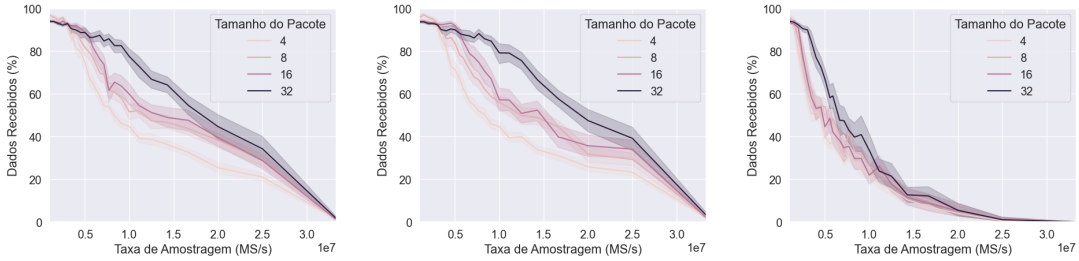
Comparando-se todos os ambientes testados com os diferentes modos de comunicação, é possível concluir que o ambiente de máquina virtual incorre no maior uso de CPU dentre os três ambientes. No entanto, para taxas de amostragem pequenas, a máquina virtual tem desempenho equivalente ao Linux nativo em relação à capacidade de recepção e transmissão, porém sempre com significativo acréscimo na utilização de CPU. Por outro lado, o contêiner apresentou desempenho semelhante ao de ambiente Linux nativo para taxas de até 25 MS/s em relação à capacidade de comunicação, assim como de uso da CPU. No que tange aos dispositivos, observa-se um melhor desempenho do B200 que apresentou melhores resultados em termos de comunicação e processamento para a maior parte dos cenários de teste.

3.2.3 Comunicação OFDM em ambientes virtualizados

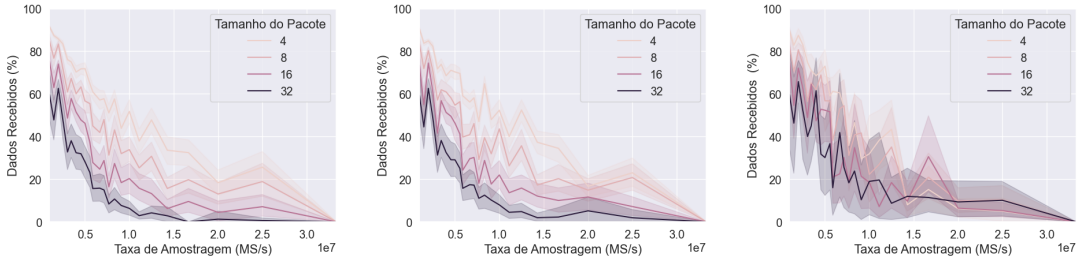
O terceiro teste realizado avalia a comunicação SDR utilizando OFDM. O objetivo desse teste é analisar o impacto que os parâmetros de comunicação, os ambientes de virtualização e os diferentes dispositivos SDR impõem no desempenho da comunicação. A aplicação foi desenvolvida usando a ferramenta GNURadio e, ao executá-la nos cenários de teste, observa-se uma perda recorrente das amostras da comunicação. Devido a essa limitação, investiga-se como os parâmetros de comunicação e o modelo USRP impactam nessa perda de dados. Para isso, implementa-se um sistema OFDM usando GNURadio que envia e recebe um arquivo texto. Os arquivos enviados e recebidos são comparados e verifica-se quantos caracteres estão incorretos ou faltando. Avalia-se a influência da variação de alguns parâmetros na qualidade da comunicação. O primeiro parâmetro é o tamanho do pacote, variando de 4 a 32 Bytes, o segundo é o número de portadoras, variando de 8 a 48 Bytes e o terceiro é a taxa de amostragem. Os ambientes, dispositivos e taxas de amostragem são idênticos àqueles do teste anterior. Os experimentos foram executados dez vezes para cada combinação de ambiente, dispositivo USRP, tamanho do pacote e número de portadoras. Os resultados são os valores médios apresentados dentro de um intervalo de confiança de 95%.

A intenção deste teste é avaliar os dispositivos da maneira mais isolada possível, ignorando fontes externas de interferência que podem influenciar as medições. Portanto, é utilizada uma configuração cabeada, de forma a reduzir a influência do ruído ambiente que estaria presente em uma comunicação sem fio. Assim, são utilizados dois dispositivos SDR conectados entre si através de cabos com atenuadores. A atenuação do sinal de saída é importante para atender as especificações dos sinais de entrada do dispositivo, uma vez que a perda de sinal provocada pelo cabo é muito pequena e sinais de potência elevada podem danificar o equipamento.

A Figura 3.9 indica a porcentagem de caracteres recebidos corretamente em todos os cenários testados para diferentes tamanhos de pacotes. É possível notar que o dispositivo B200 apresenta o comportamento mais previsível e tem um desempenho melhor que o dispositivo N200 em todos os ambientes. O dispositivo B200 obtém o melhor resultado com um tamanho de pacote de 32 bytes, porém há um declínio no desempenho à medida que o tamanho do pacote diminui, conforme mostrado nas Figuras 3.9 (a), (b) e (c). Para o dispositivo N200, quanto menores são os pacotes, melhor é a comunicação, conforme ilustrado nas Figuras 3.9 (d), (e) e (f). O número de caracteres recebidos com sucesso diminui com o aumento da taxa de amostragem. Vale ressaltar que o comporta-



(a) B200 - Linux nativo. (b) B200 - Contêiner. (c) B200 - Máquina virtual.



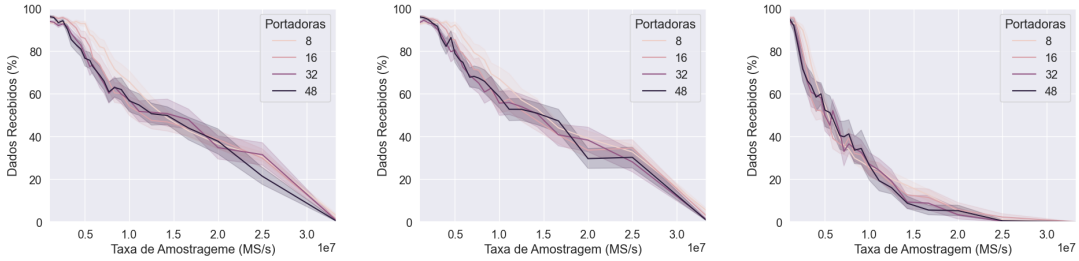
(d) N200 - Linux nativo. (e) N200 - Contêiner. (f) N200 - Máquina Virtual.

Figura 3.9: Dados recebidos de acordo com a taxa de amostragem e o tamanho do pacote. Figuras (a), (b) e (c) mostram os resultados para o USRP B200 em ambientes nativos do Linux, contêiner e máquina virtual, respectivamente. Analogamente, as Figuras (d), (e) e (f) apresentam os resultados para o USRP N200. Esses números demonstram que as aplicações que utilizam os USRPs com máquina virtual têm o pior desempenho na maioria dos cenários.

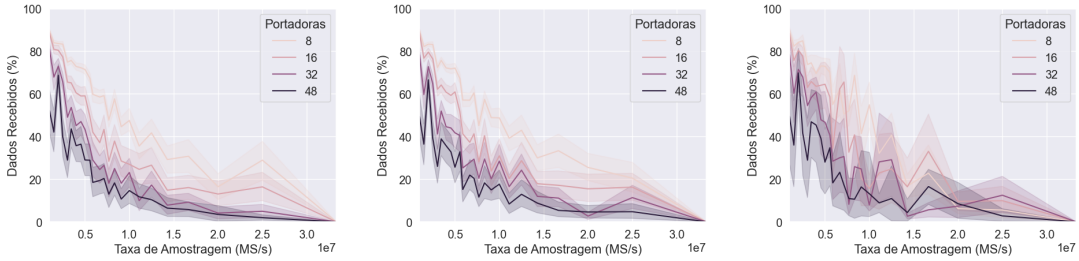
mento dos dispositivos USRP é muito semelhante nos ambientes Linux nativo e contêiner. No entanto, dentro do ambiente da máquina virtual, o desempenho do sistema diminui sistematicamente.

A Figura 3.10 indica uma comparação similar com diferentes números de portadoras. Mais uma vez, o dispositivo USRP B200 apresenta um comportamento mais previsível e um desempenho melhor que o dispositivo N200 em todos os ambientes. O dispositivo N200 apresenta um desempenho claramente distinto para um número diferente de portadoras, obtendo melhores resultados para menos portadoras.

A Figura 3.11 mostra o número médio de caracteres recebidos em cada ambiente em relação a todos os cenários de teste. Através da Figura 3.11(a), que mostra o desempenho do dispositivo B200, é possível observar que o Linux nativo e o contêiner apresentam um desempenho próximo entre si, porém a máquina virtual apresenta um resultado significativamente inferior. Para o N200, observa-se, através da Figura 3.11(b), que todos os ambientes têm desempenho semelhante. No entanto, a VM apresenta o pior desempenho em taxas de amostragem mais altas. Conclui-se que o Linux nativo e o contêiner pos-

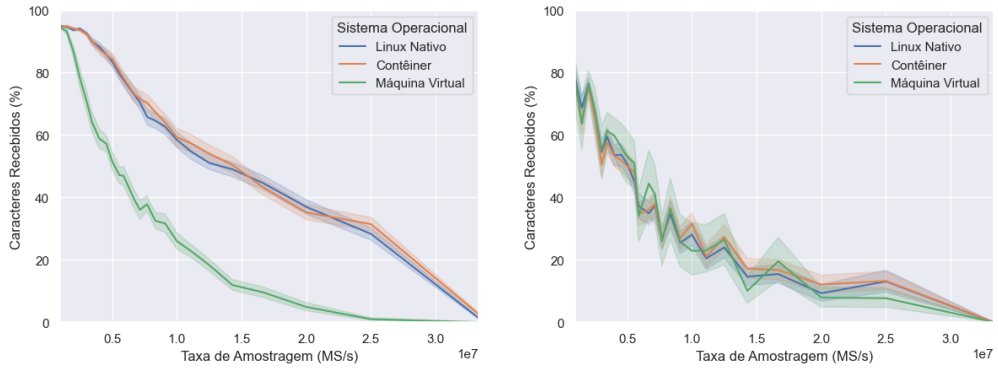


(a) B200 - Linux nativo. (b) B200 - Contêiner. (c) B200 - Máquina virtual.



(d) N200 - Linux nativo. (e) N200 - Contêiner. (f) N200 - Máquina Virtual.

Figura 3.10: Dados recebidos de acordo com a taxa de amostragem e o número de portadoras. Figuras (a), (b) e (c) os resultados para o USRP B200 em ambientes nativos de Linux, contêiner e máquina virtual, respectivamente. As Figuras (d), (e) e (f) os resultados para o USRP N200. É possível observar a diferença de comportamento de cada dispositivo em relação a variação do número de portadoras.



(a) B200. (b) N200.

Figura 3.11: Número de caracteres recebidos compilado por dispositivo. (a) para B200, a Máquina Virtual tem o pior desempenho. (b) para N200, o desempenho é semelhante nos três ambientes. Comparando as duas figuras, é possível verificar que o B200 apresenta o melhor desempenho.

suem comportamento similar independente do USRP. Comparando os dois dispositivos, é possível observar que o dispositivo B200 apresenta um desempenho superior ao N200 em todos os cenários testados.

3.3 Análise de Segurança

O uso de novas tecnologias nas redes 5G, como SDR, processamento em nuvem e virtualização, introduz novas questões de segurança e privacidade com riscos que ainda não são completamente conhecidos [28]. É importante que essas novas tecnologias sejam utilizadas de forma a extrair o máximo de suas funcionalidades, porém sem comprometer a segurança do sistema como um todo.

Além disso, apesar do uso de um computador de uso geral, as preocupações de segurança de um sistema SDR diferem de um computador pessoal devido à capacidade de um dispositivo SDR corrompido interferir e prejudicar outras redes sem fio. Assim, em relação aos cenários de teste deste trabalho, pode-se classificar as ameaças à segurança em dois grupos distintos. A primeira compreende ameaças relacionadas a *hardware* SDR e a ferramenta GNU Radio. O segundo grupo abrange ameaças ao computador hospedeiro, contêineres e máquinas virtuais.

As principais ameaças de segurança decorrentes da tecnologia SDR são o uso não-autorizado do *hardware*, reconfiguração maliciosa do SDR e falha de *software* [29]. A reconfiguração dos parâmetros de comunicação, como modulação, frequência e potência de saída, são suficientes para a realização de um ataque de *Denial of Service* (DoS) contra outros dispositivos. A mudança na frequência de operação pode levar a uma transmissão em uma frequência licenciada, interferindo no sinal autorizado e interrompendo a comunicação de diferentes redes [30]. Além disso, quando um dispositivo SDR opera com maior potência de saída, ele pode forçar outros nós a aumentar sua potência de saída, aumentando seu consumo de energia e, potencialmente, drenando suas baterias. Outra fonte de vulnerabilidades é o GNU Radio, que permite a instalação de novos módulos complementares à sua biblioteca padrão. Embora esses módulos possam reconfigurar as funções do SDR, o GNU Radio não verifica se o módulo contém código malicioso ou se a fonte do módulo é confiável [31].

As tecnologias de virtualização também introduzem várias ameaças e problemas de segurança em um sistema de comunicação. O ambiente de compartilhamento de recursos traz novas ameaças, uma vez que os sistemas operacionais virtualizados podem ser mal-intencionados e realizar ataques a outros sistemas convidados, assim como pode atacar o computador hospedeiro ou o VMM [32]. Além disso, as imagens no repositório Docker Hub não estão livres de vulnerabilidades. Desikan afirma que mais de 30% das imagens oficiais no Docker Hub contêm vulnerabilidades de segurança de alta prioridade [33]. Shu

et al. descobriram que, em média, as imagens oficiais e da comunidade contêm mais de 180 vulnerabilidades, e as imagens filhas trazem cerca de 20 novas vulnerabilidades [34].

Em relação às vulnerabilidades SDR e GNU Radio, Hill et al. propõem limitar a permissão para reconfigurar um SDR para agentes confiáveis e autorizados [31]. Eles também sugerem que o dispositivo SDR deve ser capaz de verificar se está executando o código de aplicativo apropriado e que uma configuração orientada por política deve existir para restringir e orientar a operação do SDR. Para minimizar as vulnerabilidades de virtualização, Sierra et al. propõem usar ferramentas que garantam isolamento de domínio, separação de privilégios e segurança baseada em virtualização, como o Virtual Secure Mode (VSM). As vulnerabilidades associadas às imagens do Docker Hub podem ser minimizadas usando as imagens oficiais mais recentes para implantar contêineres e criar novas imagens filhas [32].

3.4 Conclusão sobre SDR em ambientes virtualizados

Esta dissertação avalia os limites práticos dos dispositivos SDR e o impacto da virtualização no recebimento e envio de amostras, bem como em uma aplicação OFDM. Primeiro, testa-se a taxa de amostragem na qual é possível transmitir e receber amostras com os dispositivos USRP B200 e N200. Em seguida, utilizam-se três ambientes diferentes: Linux nativo, contêiner e máquinas virtuais, com três cenários de comunicação diferentes: transmissão simplex, recepção simplex e comunicação full-duplex. No geral, o ambiente VM teve um desempenho pior do que todos os outros ambientes testados, apresentando o maior consumo de CPU. Por outro lado, os resultados mostram que, para a maioria dos cenários testados, o ambiente contêiner obteve resultados equivalentes ao ambiente Linux nativo.

A comunicação OFDM em dispositivos USRP foi analisada com diferentes tamanhos de pacotes e números de operadoras. Os resultados mostram que os dois dispositivos se comportam de maneira diferente quanto à variação de cada parâmetro, porém o dispositivo B200 se comunica com a menor perda de dados. Com relação ao desempenho em ambientes virtualizados, não é observada uma degradação significativa da comunicação devido à virtualização, exceto para a aplicação SDR em VM com o dispositivo B200. Os dois dispositivos USRP tiveram comportamentos, desempenho e limitações distintas, o que é explicado pela diferença em sua arquitetura, capacidade dos seus componentes, como FPGA e conversores A/D e D/A, e também no modo de conexão com o computador.

Como avaliação geral do uso os USRPs em ambientes virtualizados, observa-se que o uso de 30% de CPU é capaz de influenciar negativamente as taxas de comunicação das aplicações SDR. Assim, no desenvolvimento de aplicações de comunicações em ambientes virtualizados, é importante se ater a taxas de amostragem e modos de comunicação que não sobrecarreguem a máquina física, com o intuito de evitar perdas de pacotes, e utilizar protocolos de camadas superiores que prevejam retransmissão de dados. Além disso, é importante adotar medidas de segurança, como instalação de módulos seguros no GNU Radio e imagens de repertórios confiáveis, de maneira a diminuir a vulnerabilidade das aplicações.

Capítulo 4

Privacidade em IoT e Modelos de Aprendizado de Máquina

As redes móveis de 5ª Geração visam fornecer desempenho superior às gerações anteriores em termos de banda larga móvel, comunicações massivas do tipo máquina (mMTC) e latência [35]. Um dos principais casos de uso da rede 5G são as comunicações massivas do tipo máquina-a-máquina, projetadas para suportar a conexão de muitos dispositivos IoT, como sensores, *wearables* inteligentes, eletrodomésticos e máquinas industriais. Os dispositivos IoT estão cada vez mais presentes nas atividades cotidianas e, impulsionado por essa disseminação de dispositivos, o número de ataques também cresceu intensamente, com expansão de mais de 100% no primeiro semestre de 2021 [2]. Dessa forma, a conexão autônoma que as redes 5G oferecem a esses dispositivos levanta novas preocupações de segurança e privacidade que devem ser abordadas, considerando as restrições dos dispositivos IoT em se proteger contra ataques.

4.1 Segurança e privacidade de dispositivos IoT

Os dispositivos IoT apresentam especificações de energia, processamento, rede e armazenamento que diferem significativamente dos nós tradicionais da Internet, como servidores, computadores e notebooks [3]. Enquanto isso, devido à diversidade de propósitos e designs de fabricação, os dispositivos IoT também são heterogêneos, dificultando uma abordagem universal para proteger a conexão dos dispositivos. Além disso, devido à demanda por dispositivos de baixo custo, o projeto dos componentes não contempla soluções que assegurem a segurança e privacidade dos usuários. Como consequência, os dispositivos IoT manifestam capacidade restrita para resistir a ataques cibernéticos [4]. Além disso, de-

vido à demanda por dispositivos de baixo custo, o design dos fabricantes negligencia as questões de segurança e privacidade.

A onipresença de dispositivos IoT levanta preocupações com a privacidade, uma vez que esses dispositivos são incorporados com recursos de comunicação e interagem com usuários ou servidores remotos. A atividade de rede dos dispositivos pode indicar comportamentos e rotinas dos usuários, mesmo sem acesso à carga útil do tráfego de rede [36, 37]. Os usuários não sabem que os dispositivos IoT coletam informações, mesmo quando não são usados ativamente e podem enviar dados para terceiros [38]. A proposta desta dissertação se concentra na representação estatística de dados de rede para permitir o reconhecimento preciso dos dispositivos conectados a determinada rede.

4.2 Aprendizado de Máquina

O aprendizado de máquina é uma etapa da inteligência artificial na qual o computador é capaz de aprender com os dados, inferir parâmetros de um modelo e tomar decisões. Há uma variedade de tarefas que um algoritmo de aprendizado de máquina pode realizar, como regressão ou ranqueamento, porém este trabalho utiliza somente a capacidade de classificação dos algoritmos. Em um sentido amplo, o processo de aprendizado pode ser categorizado em aprendizado supervisionado, não-supervisionado e por reforço.

No aprendizado supervisionado, há um modelo que representa algumas funções matemáticas combinadas capazes de predizer uma saída y a partir de uma entrada x. As funções podem possuir diversos parâmetros que precisam ser ajustados, através de aprendizado, para que o modelo consiga desempenhar a tarefa desejada. Assim, para o modelo adquirir a habilidade de predizer as saídas y corretas, é necessário existir uma etapa prévia de treinamento. Nessa etapa, são apresentadas diversas entradas ao modelo, cujas saídas já são conhecidas e são chamadas de rótulos, e o modelo ajusta os seus parâmetros, como peso e/ou topologia, para se aproximar da respectiva saída desejada, através de correção de erro.

No aprendizado não-supervisionado, não há exemplos rotulados ou pares entrada-saída para realizar o treinamento prévio da rede. O algoritmo agrupa os dados de acordo com suas similaridades, encontrando padrões e correlações de acordo com as características dos dados. No aprendizado por reforço, o mapeamento de entrada-saída é realizado através da interação contínua com o ambiente, visando minimizar um índice de desempenho. Esse tipo de rede possui a habilidade de aprender a realizar uma tarefa predeterminada com

base apenas nos resultados de sua própria experiência.

Um importante conceito de aprendizado de máquinas é a generalização que consiste na capacidade de um modelo de avaliar e tratar uma entrada não conhecida, produzindo uma resposta adequada. Um modelo generaliza bem quando o mapeamento de entrada-saída realizado é correto. Para avaliar o quão bem um modelo está ajustado ao conjunto de dados, é utilizada uma função objetivo composta de uma função de erro somada a um termo regulador. Uma das funções de erro mais comuns é o erro médio quadrático: $E = \sum_i (y_i - \hat{y}_i)^2$, onde y_i é o valor do rótulo e $\hat{y}_i$ é o valor estimado pelo modelo. Durante a etapa de treinamento, o modelo irá ajustar todos os seus parâmetros para minimizar a função erro E. Porém, quando um modelo aprende muitos exemplos de entrada-saída, é possível que memorize os exemplos de treinamento e perca a sua capacidade de generalizar, o que é conhecido como sobreajuste (*overfitting*). O termo regulador tem como objetivo reduzir a complexidade do modelo, penalizando determinadas configurações e, com isso, reduzindo o sobreajuste. A generalização depende dos seguintes fatores: a quantidade de exemplos de treinamento e quão representativos são estes exemplos em relação ao ambiente de interesse, a arquitetura do modelo e a complexidade física do problema a ser resolvido.

A validação cruzada é uma ferramenta comum em estatística, na qual o conjunto de dados é particionado aleatoriamente em conjunto de treino e conjunto de teste. As amostras de treinamento são ainda subdivididas em subconjunto de estimação, usado para ajustar o modelo, e subconjunto de validação, usado para validar o modelo. O conjunto de teste é utilizado para avaliar o desempenho do modelo. O *k-fold* é um método de validação cruzada múltipla onde um conjunto de dados de N exemplos é dividido em k grupos. O modelo é treinado com todos os grupos, com exceção de um, utilizado para etapa de validação e cálculo do erro de validação. Esse procedimento é repetido k vezes e, a cada iteração, utiliza um grupo diferente para a validação. O desempenho do modelo é avaliado pelo erro médio quadrático obtido nas k execuções.

Visando reduzir o sobreajuste, é possível monitorar uma métrica específica do treinamento do modelo e pará-lo, caso seja identificada uma tendência de aprendizado excessivo do conjunto de treinamento ou uma estagnação no aprendizado. Esse método é chamado de parada antecipada e, neste trabalho, é realizado através do monitoramento do erro de estimação do conjunto de validação. Com o passar das épocas e a evolução do treinamento, a capacidade de generalização do modelo tende a melhorar e o erro de estimação a diminuir. Porém, se o erro do conjunto de treinamento diminui e o erro do conjunto

de validação aumenta ou fica estagnado após determinadas épocas, o modelo está se especializando no conjunto de treinamento e perdendo a sua capacidade de generalização. Dessa forma, continuar o treinamento tende a agravar a situação e é neste momento onde a parada antecipada é realizada. Devido à iteratividade do processo de treinamento, pode ocorrer, em determinadas épocas, uma degradação do desempenho do conjunto de validação que não necessariamente caracterizam um sobreajuste, mas sim uma etapa intermediária no processo de aprendizado. Para que seja utilizada a parada antecipada, é necessário que o erro de estimação com o conjunto de validação piore por algumas épocas para evidenciar o sobreajuste. Dessa forma, existe o parâmetro "paciência" que consiste no número de épocas sem melhora do erro de validação após as quais o treinamento será interrompido.

Há diversas abordagens em aprendizado de máquina para a tarefa de classificação. Neste trabalho, são utilizadas árvores de decisão e redes neurais. A seguir segue uma breve descrição dos modelos utilizados.

4.2.1 Árvore de decisão

A árvore de decisão de interesse para este trabalho utiliza o algoritmo de Árvores de Classificação e Regressão, *Classification and Regression Trees* (CART), que consiste em um algoritmo de aprendizado supervisionado utilizado para resolver problemas de classificação e regressão. Uma árvore é uma estrutura hierárquica composta de um nó raiz, nós de decisão e nós folhas. Para cada um dos nós de decisão, é realizado um teste de determinado atributo em relação a um limiar. Dependendo do resultado do teste, o algoritmo segue para um ou outro nó. Esse processo se repete até ser atingido um nó folha, onde são apresentados os resultados do modelo. Um passo importante do modelo de árvore é selecionar a ordem em que os testes aparecem, de forma a melhorar a capacidade preditiva do modelo. O algoritmo CART utiliza árvores binárias cujas decisões são baseadas em características e limiares definidos pelo índice Gini. Quanto maior é o tamanho da árvore, maiores são as chances do modelo ter uma determinada parte da árvore representando apenas poucas entradas, o que pode levar a um sobreajuste do modelo. Dessa forma, a profundidade da árvore é um importante fator para as árvores de decisão e deve ser ajustada com cautela, dando preferência a árvores pequenas. Assim, uma das técnicas utilizadas para evitar o sobreajuste é limitar a profundidade da árvore e a outra é chamada poda (*prunning*) que, resumidamente, exclui os ramos que testam características de baixa importância e limitam o tamanho da árvore.

XGBoost é um método de aprendizado de máquina que usa um conjunto de CART para realizar tarefas de classificação e regressão [39]. A técnica de reforço (*Boosting*) consiste em usar vários modelos simples combinados para criar um sistema de aprendizagem mais robusto. No caso do XGBoost, iterativamente, o algoritmo cria uma árvore, complementar às já existentes, e a adiciona ao modelo, com o intuito de melhorar a sua capacidade de predição.

Assim como para as CARTs empregadas isoladamente, o tamanho das árvores também é um parâmetro importante do XGBoost e pode ser configurado através do parâmetro de profundidade máxima, cujos valores mais altos aumentam a capacidade de ajustar os dados, mas, em contrapartida, aumentam a complexidade e o risco de sobreajuste.

4.2.2 Rede Neural Artificial

As rede neurais artificiais são inspiradas na maneira como o cérebro realiza tarefas, empregando uma interligação maciça de células computacionais simples, denominadas "neurônios" ou "unidades de processamento". Assim como no cérebro humano, na rede neural o conhecimento é adquirido através de um processo de aprendizagem e armazenado através da força de conexão dos neurônios, conhecidos como pesos sinápticos. O algoritmo de aprendizagem modifica os pesos sinápticos para alcançar os objetivos do projeto e, em consequência dessa etapa de aprendizagem, a rede adquire a capacidade de generalização, que consiste em produzir saídas adequadas para entradas que não estavam presentes durante a fase de treinamento.

Um neurônio é uma unidade de processamento fundamental na operação de uma rede neural. A Figura 4.1 representa o modelo neuronal básico, onde um conjunto de entradas é multiplicado pelos pesos w_{ki} que representam as sinapses. Além das entradas, há uma constante x_0, utilizada para regular o valor do campo local induzido v_k. Por último, está a função de ativação que irá fornecer a saída do neurônio em valores restritos a determinados intervalos, como entre 0 e 1 ou -1 e 1.

Uma rede neural é composta por diversos neurônios podendo ser dispostos e interligados de diferentes formas com diferentes arquiteturas. Dentre elas, três classes apresentam maior relevância para este trabalho. A primeira consiste nas redes neurais diretamente alimentadas de apenas uma camada, chamadas de perceptron de uma camada. Essas redes possuem uma camada de entrada de "nós de fonte" que se interliga em uma camada de saída de neurônios, mas não vice-versa. É uma rede considerada de camada única, referente apenas à camada de saída, visto que a camada de entrada não realiza qual-

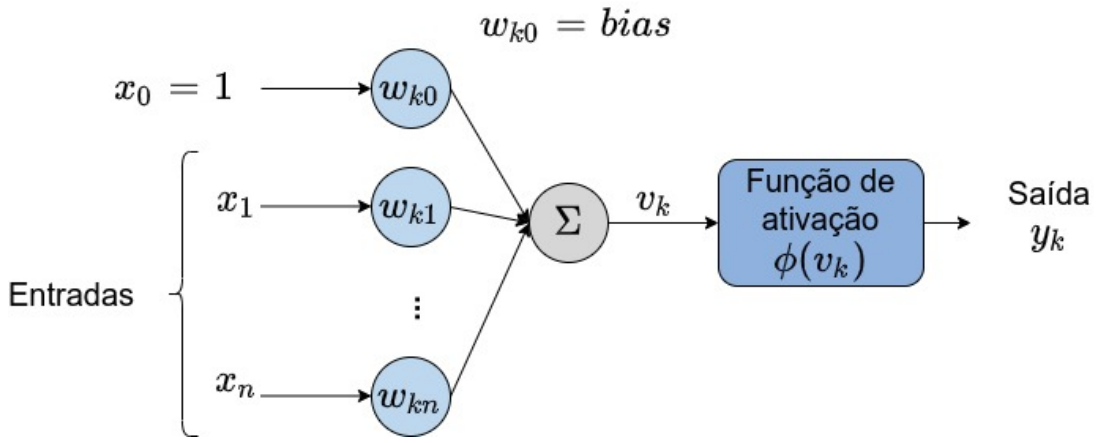


Figura 4.1: Modelo neuronal, onde as entradas são multiplicadas por pesos $w_k i$ e somadas, resultando no campo de ativação local v_k. A função de ativação $\phi(.)$ controla a saída y_k do neurônio de acordo com o valor de v_k.

quer computação. Uma segunda classe comum são os perceptrons de múltiplas camadas, ilustrados na Figura 4.2, que consistem em redes alimentadas diretamente com múltiplas camadas que, além da entrada e saída, possuem uma ou mais camadas intermediárias, chamadas de camadas ocultas. Essas camadas adicionais permitem a extração de características de ordem mais elevada pela rede. Nessa configuração, os neurônios em cada camada da rede têm como entradas apenas os sinais de saída da camada precedente. Por último, tem-se as redes recorrentes que se diferem das arquiteturas anteriores por terem pelo menos um laço de realimentação, ou seja, o sinal de saída de um ou mais neurônios alimenta a entrada da própria camada ou das anteriores. O caso em que a entrada de um neurônio é realimentada pela sua própria saída é chamado de auto-realimentação. Independentemente das características mencionadas anteriormente, uma rede neural é considerada totalmente conectada quando algum nó de uma camada da rede está conectado a todos os nós da camada seguinte. Na ausência de alguma ligação, a rede é dita parcialmente conectada.

O treinamento de um perceptron de múltiplas camadas é realizado através do algoritmo de retropropagação (*backpropagation*) que é dividido em duas etapas: a primeira, chamada de fase adiante (*forward pass*), ocorre calculando as saídas da rede a partir da entrada. A segunda, chamada de (*backward pass*), acontece no sentido inverso calculando o gradiente da função erro e atualizando os pesos da rede neural. Dessa forma, são observados dois sinais: sinais de função, que correspondem ao sinal de entrada se propagando camada-a-camada até a saída da rede, e sinais de erro, que são sinais originados na saída da rede e se propagam no caminho inverso até a entrada, camada por camada.

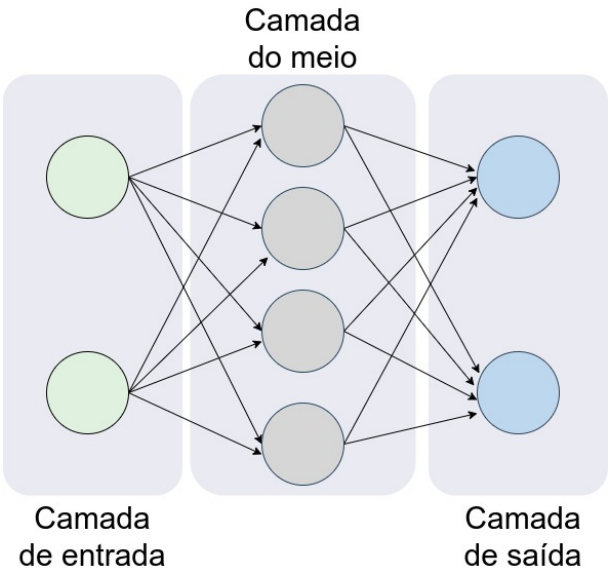


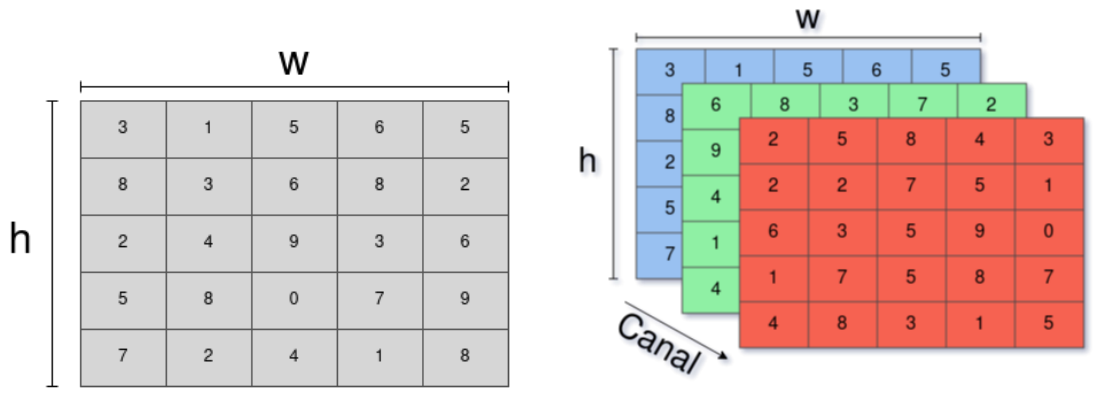
Figura 4.2: Rede Neural Artificial totalmente conectada com uma camada de entrada, uma camada escondida e uma camada de saída.

Neste trabalho, as redes neurais são treinadas usando um conjunto de treinamento subdivididos em lotes. Dessa forma, os pesos sinápticos da rede só são recalculados e atualizados depois da passagem de um lote contendo N amostras de treinamento. Além disso, a rede é treinada por diversas épocas. Assim, a cada época, o conjunto de treinamento é embaralhado antes de ser apresentado novamente à rede.

4.2.2.1 Rede Neural Convolucional

A CNN é um mecanismo de aprendizado de máquina amplamente utilizado na classificação de imagens. As entradas de uma CNN podem ser interpretadas como imagens que são tensores tridimensionais, onde duas dimensões representam a largura w e a altura h da imagem e a terceira dimensão representa os canais de cores c. Uma imagem em um espaço de cores RGB contém três canais de cores, vermelho, azul e verde, enquanto uma imagem em escala de cinza tem apenas um canal. Os *pixels* são valores numéricos que representam a intensidade da cor em determinado ponto. A diferença entre a representação dos dois tipos de imagens é ilustrada na Figura 4.3.

O modelo CNN baseia-se no conceito da operação matemática conhecida como convolução bidimensional, onde um filtro matricial de dimensão $n \times n$ percorre a imagem para extrair características da imagem. Em uma camada de convolução, o filtro é aplicado a uma área da imagem, chamada de campo receptivo local, então o produto escalar entre o filtro e os *pixels* é calculado e inserido em uma matriz. A matriz de saída da camada



(a) Imagem em escala de cinza.

(b) Imagem colorida.

Figura 4.3: Comparação entre a representação de (a) uma imagem em escala de cinza e (b) uma imagem colorida.

de convolução é conhecida como mapa de características ou mapa de ativação. A primeira camada de convolução é capaz de detectar características simples de uma imagem, como bordas e cor, porém, as camadas subsequentes vão aumentando a capacidade de reconhecer características mais complexas, como pessoas ou animais.

Além das camadas de convolução, a CNN utiliza camadas de *pooling* e camadas totalmente conectadas. A camada de *pooling* é utilizada após uma camada de convolução, sendo responsável por reduzir as dimensões do mapa de características com o objetivo de diminuir o custo computacional de processamento. Para isso, um *kernel* de dimensões $m \times m$ percorre com passo (*stride*) p o mapa de características, agregando os valores e gerando uma matriz de saída com dimensões menores que a matriz de entrada. Há dois métodos principais para agregação dos valores: o primeiro, conhecido como *Max Pooling*, utiliza o valor máximo da região $m \times m$ do mapa de características percorrida pelo *kernel*; e o segundo, conhecido como *Average pooling* utiliza a média desses valores. Os dois métodos são ilustrados na Figura 4.4, onde é possível observar que o Max *Pooling* extrai as características mais proeminentes do mapa de características, enquanto o *Average pooling* suaviza os valores mais extremos, resultando em uma imagem mais suave.

O resultado da camada de *pooling* são múltiplos mapas de características. O passo seguinte em uma CNN é realizar o achatamento desses mapas em um vetor, como ilustrado na Figura 4.5. Esse vetor consiste na camada de entrada de uma rede neural artificial.

Os próximos passos são semelhantes a uma rede neural artificial, com uma camada de entrada, uma camada oculta, e uma camada de saída, conforme ilustrado na Figura 4.2. No contexto de uma CNN, a camada escondida é chamada de camada totalmente conec-

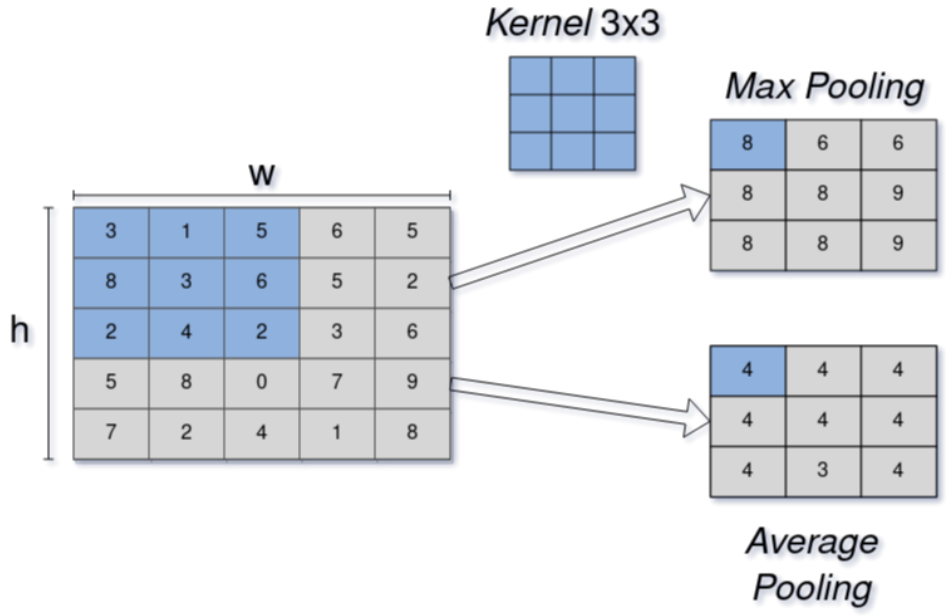


Figura 4.4: Exemplo da camada de *pooling* usando *kernel* 3x3 e passo 1. É possível visualizar a saída da camada utilizando as técnicas de Max e *Average pooling*.

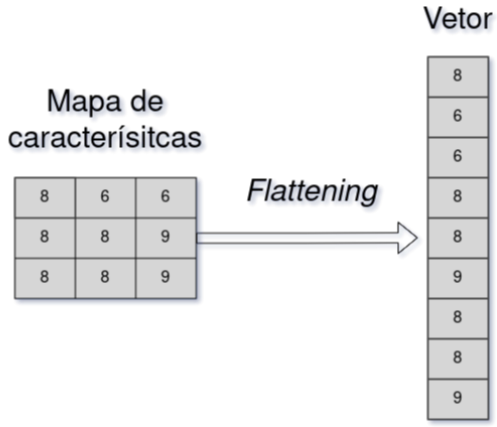


Figura 4.5: Exemplo do achatamento de um mapa de recursos em vetor em uma rede CNN.

tada. A camada de saída possui uma quantidade de nós equivalente ao número de classes a serem classificadas e o resultado da classificação representa a probabilidade da imagem de entrada pertencer àquela classe específica.

4.2.3 LSTM

Uma Rede Neural Recorrente (RNN) é uma classe de rede neural capaz de aprender relações temporais ou sequenciais e possui alguns laços, ou seja, o processamento da rede

considera não só as entradas atuais como entradas anteriores. Dessa forma, as redes possuem "memória" e utilizam estados anteriores para realizar as previsões futuras. Esse recurso permite que as RNNs aprendam as relações temporais entre os dados, tornando o modelo popular em aplicações com dados variáveis no tempo, como reconhecimento de voz, tradução de idiomas e previsão de séries temporais.

As RNNs consistem em diversos módulos, chamados células, encadeados um após o outro. Nesse tipo de rede, é usado o algoritmo de retropropagação através do tempo para calcular os gradientes e ajustar o peso dos nós da rede. Assim, cada passo de tempo é considerado como uma camada. A desvantagem do método é que o gradiente desaparece exponencialmente à medida em que se propaga pelas camadas, fazendo com que as primeiras camadas sofram sempre um pequeno ajuste e, consequentemente, tenham menor capacidade de aprendizado. Assim, as RNNs tradicionais são consideradas redes de memória de curta duração.

A rede LSTM é uma RNN especializada capaz de aprender dependências de longo prazo. O modelo é composto por unidades de memória, cada uma contendo três portas, uma entrada, uma saída e uma porta de esquecimento. A porta de entrada é responsável por carregar os valores da entrada na memória. A porta de saída decide quais informações de memória são apresentadas como entrada para o mecanismo de aprendizado do nó. Por fim, a porta de esquecimento controla quais informações são descartadas da memória. Assim, a unidade de memória lembra de valores por um intervalo de tempo, de acordo com os controles realizados pelo conjunto das portas.

Existem algumas variações de arquiteturas de células LSTM e a versão utilizada nesse trabalho está ilustrada na Figura 4.6 com o seu funcionamento está descrito a seguir. A entrada x_t de uma célula e a saída da célula anterior h_{t-1} são combinadas e apresentadas na entrada das três portas da célula. O primeiro passo na rede LSTM é decidir o que manter ou excluir no estado da célula. Essa etapa é controlada pelo porta de esquecimento que utiliza uma função sigmoide σ para decidir se cada item do estado da célula anterior c_{t-1} será mantido ou esquecido no estado da célula atual c_t. O segundo passo consiste em decidir quais informações serão adicionados à célula. A porta de entrada, também através da função σ, define quais os valores de c_t serão atualizados com novos dados, a função $tanh$ gera esses novos valores que serão somados aos valores de c_{t-1} não esquecidos, gerando o estado da célula atual c_t. O último passo é o cálculo da saída h_t. A função $tanh$ calcula os valores de saída através do estado da célula c_t e a porta de saída, através da função σ, decide quais desses dados irão compor a saída h_t.

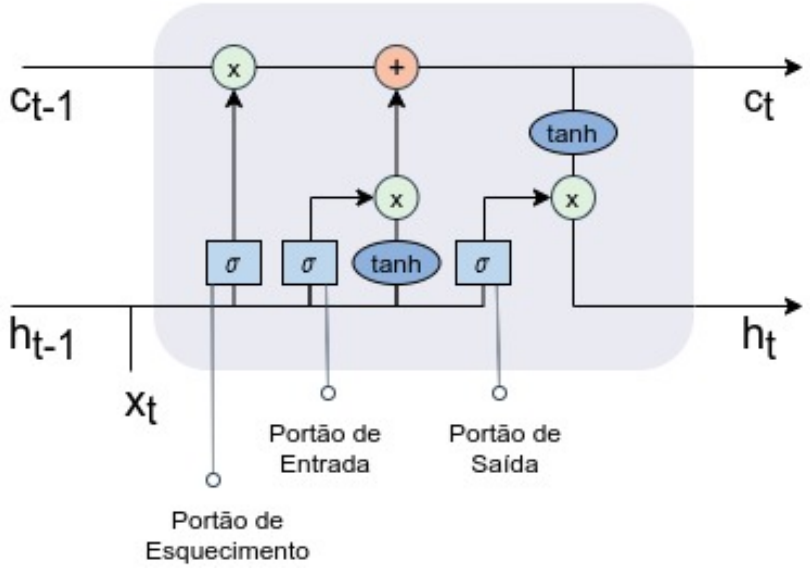


Figura 4.6: Arquitetura de uma célula LSTM. A saída da célula anterior h_t é combinada com uma entrada x_t e passada à portas da célula atual. A porta de esquecimento define quais informações do estado da célula anterior será armazenado. O porta de entrada define quais os campos de c_t receberão novos dados. O porta de saída estabelece quais dados do estado c_t aparecerão na saída h_t.

4.3 Trabalhos relacionados

Trabalhos anteriores se concentram na classificação de dispositivos IoT e discutem abordagens para melhorar a precisão no reconhecimento de dispositivos IoT. Sivanathan *et al.* visam identificar dispositivos IoT específicos através de um método de classificação em dois estágios [40] com base em sua atividade de rede. O primeiro estágio de classificação implanta o algoritmo de Bayes para classificar dispositivos com base em domínios contatados, em portas de comunicação e em conjuntos de cifras criptográficas compartilhados durante o *handshake* TLS. A classificação do segundo estágio combina os resultados do primeiro estágio com mais características, como duração, volume, taxa de fluxo, intervalos de solicitação de DNS e intervalos de solicitação de NTP. O método proposto atinge 99,88% de acurácia na identificação dos dispositivos usando janelas de tempo de uma hora. Uma desvantagem da abordagem é que a janela de tempo é longa para a maioria dos aplicativos de classificação de dispositivos IoT, como em detecção de intrusão ou gerenciamento de rede, além de se basear em números de portas e nomes de domínios que podem ser alterados facilmente em atualizações futuras. Shahid *et al.* propõem a identificação de dispositivos IoT utilizando informações dos fluxos de rede dos dispositivos, como tamanho do pacote e tempo entre chegadas de pacotes [41]. A abordagem proposta classifica os dispositivos com 99,9% de precisão, mas a análise implementada considera

apenas quatro dispositivos com comportamento claramente distinto: uma câmera, uma lâmpada, um sensor e um plugue. Além disso, Meidan *et al.* identificam dispositivos IoT com 99,28% de precisão classificando características extraídas de sessões TCP [42]. No entanto, essa abordagem é limitada, pois alguns dispositivos IoT não usam o protocolo TCP. Por sua vez, Abbas *et al.* identificação de dispositivos IoT usando o algoritmo *K-Nearest Neighbors* e obtém uma acurácia média de 95% e um F1-score de 91% [36].

Os trabalhos listados a seguir realizam classificações baseadas na análise do tráfego de rede, porém com objetivos diferentes que a identificação de dispositivos. Apesar da diferença de objetivos, essas propostas são relevantes para este trabalho pelos métodos de extração de características do conjunto de dados e pelos algoritmos de classificação empregados. Bai *et al.* propõem um método automático para identificação de categorias de dispositivos IoT. O conjunto de dados contém 16 dispositivos de quatro categorias: *hubs*, eletrônicos, câmeras e chaves. O método proposto agrupa os pacotes em janelas de tempo e extrair características do tráfego de rede, como o número de pacotes recebidos e transmitidos e contagens de pacotes para diferentes protocolos [43]. O trabalho combina LSTM e CNN para atingir uma acurácia média de 74,8% no reconhecimento das categorias dos dispositivos. Da mesma forma, Lopez-Martin *et al.* propõem diferentes combinações de redes CNN e LSTM para detectar os serviços utilizados em um fluxo de rede IP [44]. A entrada do método proposto é uma matriz 20×6 representando 20 pacotes com seis características, portas de origem e destino, número de bytes, tamanho da janela TCP, tempo entre chegadas e direção do fluxo. O método proposto atinge 96% de precisão e 95% de F1-score.

Ren *et al.* monitoram o tráfego de rede de 81 dispositivos IoT, localizados nos Estados Unidos e no Reino Unido, com o objetivo de verificar a exposição de informações no tráfego de rede desses dispositivos. Os autores avaliam se o destino do tráfego são empresas terceiras que podem monitorar informações sobre os usuários, se o tráfego é criptografado, se o dispositivo envia informação de forma contínua e se o país de localização do dispositivo altera o seu comportamento na rede. O trabalho conclui que mais da metade dos destinos das comunicações são para empresas que não são os fabricantes e identifica dispositivos de gravação de áudio e video com atividade de rede inesperada [38]. A proposta desta dissertação é analisada e avaliada com base nesse conjunto de dados.

Em resumo, realizando uma comparação com os trabalhos relacionados que identificam os dispositivos IoT com base na análise do tráfego de rede, encontram-se as seguintes diferenças: a proposta desta dissertação alcança alta acurácia e precisão na classificação

de dispositivos IoT, com uma resolução de fluxo, utilizando uma janela de tempo curta, de 60 segundos, adequada para aplicações *online*. Além disso, o modelo utiliza apenas características estatísticas de fluxo, que permitem a classificação de dispositivos independentemente do protocolo da camada de aplicação empregado ou do uso de criptografia, como TLS e HTTPS. Além disso, o método é flexível o suficiente para distinguir corretamente dispositivos semelhantes, pois a análise considera um conjunto de dados de mais de 30 dispositivos, pertencentes ao conjunto de dados desenvolvido por Ren *et al.* [38], com seis categorias de dispositivos e diferentes modelos de dispositivos para cada categoria.

Capítulo 5

Reconhecimento de dispositivos IoT

Esta dissertação propõe um método para reconhecer dispositivos IoT através do monitoramento de tráfego de rede e extração de informações estatísticas do fluxo de dados que sejam capaz de identificar os dispositivos IoT. A estrutura de representação de dados proposta é um tensor que captura as correlações entre os fluxos da rede. O reconhecimento dos dispositivos é realizado através de algoritmos de classificação baseados em aprendizado de máquina, usando três abordagens diferentes, CNN, LSTM e XGBoost. Cada um dos modelos de aprendizado de máquina mencionados são testados usando diferentes configurações com o intuito de selecionar os melhores parâmetros e a arquitetura mais adequada.

5.1 Caracterização do conjunto de dados IoT

O conjunto de dados utilizado foi proposto e disponibilizado por Ren et al. [38][1], englobando dois conjuntos de experimentos, um realizado nos EUA e outro realizado no Reino Unido. No referido trabalho, os autores buscam identificar a diferença no comportamento dos dispositivos quando adquiridos e utilizados em países diferentes. Assim, os modelos utilizados em ambos os países são, em sua maior parte, semelhantes e, neste trabalho, foi utilizado apenas o subconjunto do Reino Unido para a validação desta proposta. A tabela 5.1 resume os 34 dispositivos monitorados e suas categorias. Os dados foram capturados usando `tcpdump`[2] e armazenados como arquivos `pcap`. O conjunto de dados é composto pelo tráfego gerado por meio de vários experimentos, divididos em três categorias, inicialização, interação e ociosidade. O tráfego do experimento de inicializa-

[1]Disponível em https://moniotrlab.khoury.northeastern.edu/publications/imc19/.
[2]Disponível em https: //www.tcpdump.org/.

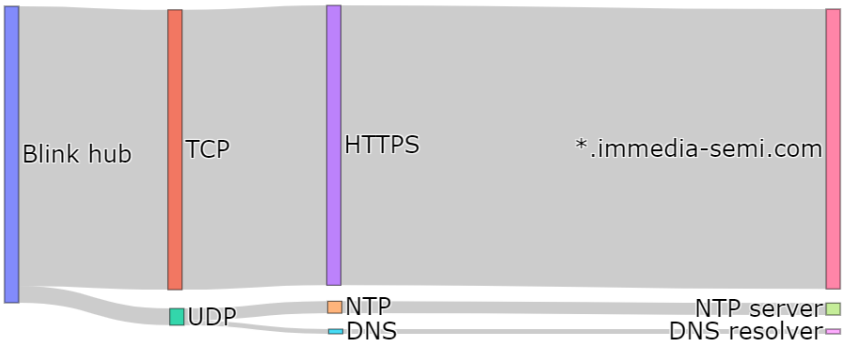
ção contém pacotes de rede coletados por dois minutos a partir do momento em que o dispositivo é ligado. Os experimentos de interação geram tráfego por meio da interação do usuário com o dispositivo, por exemplo, através de comandos de voz ou aplicativos móveis complementares. O tráfego dos dispositivos ociosos é coletado quando nenhum usuário está interagindo com o dispositivo, duram oito horas e são realizados três vezes. Os experimentos de inicialização duram dois minutos e são repetidos, no mínimo, três vezes para cada dispositivo. A duração dos experimentos de interação varia de acordo com o tipo de interação empregada e a sua repetibilidade varia de acordo com a viabilidade de automação do teste. As interações automatizadas, como o uso de aplicativo complementar e comando de voz, foram realizadas pelo menos trinta vezes, enquanto as não-automatizadas, como as interações físicas, foram realizadas ao menos três vezes.

Tabela 5.1: Categorias e modelos de dispositivos IoT monitorados.

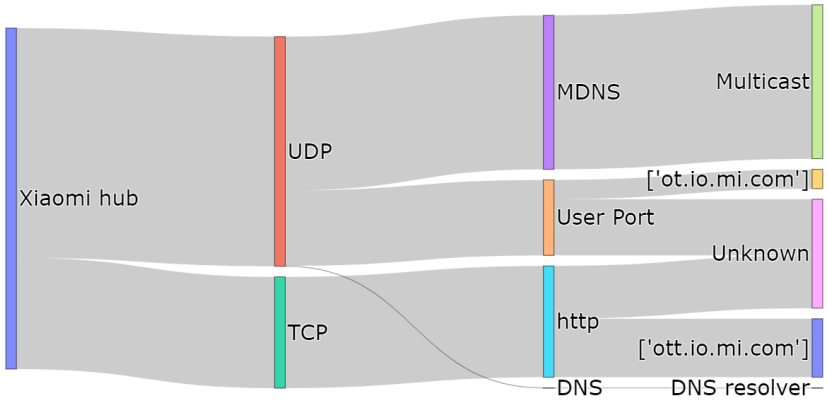
Câmeras	*Smart Hubs*	Automação Residencial
Blink Cam	Blink Hub	WeMo Plug
Bosiwo Cam	Insteon	Honeywell T-stat
Ring Doorbell	Lightify	Magichome Strip
Spy Camera	Philips Hub	Nest T-stat
Wansview Cam	Sengled	TP-Link Bulb
Xiaomi Cam	Smartthings	TP-Link Plug
Yi Cam	Xiaomi hub	
TV	**Áudio**	**Eletrodomésticos**
	Allure speaker	
Apple TV	Echo Dot	Anova Sousvide
Fire TV	Echo Spot	Netatmo Weather
Roku TV	Echo Plus	Smarter Brewer
Samsung TV	Google Home Mini	Xiaomi Cleaner
	Google Home	

5.1.1 Caracterização do tráfego

Dada a diversidade de propósitos e fabricantes, as características de comunicação dos dispositivos variam consideravelmente em termos de protocolos de aplicação, portas de comunicação, servidores e métricas estatísticas de fluxo. Para ilustrar essas diferenças, as Figuras 5.1(a) e 5.1(b) mostram o diagrama Sankey da atividade de rede de dois *hubs* de dispositivos IoT: o Blink Security Hub e o Xiaomi Hub. A Figura 5.1(a) mostra que a maioria dos pacotes do Blink Security Hub usa o protocolo de transporte TCP e HTTPS como protocolo de aplicação, usando UDP para resolver nomes de *host* através do protocolo DNS e sincronizar relógios pelo protocolo NTP. Além disso, a diversidade de



(a) Diagrama de tráfego de rede para o Blink Security Hub.



(b) Diagrama de tráfego de rede para um Xiaomi Hub.

Figura 5.1: Comparação das atividades de rede de diferentes dispositivos. Embora ambos os dispositivos sejam *hubs* inteligentes, (a) se conecta a vários servidores NTP e usa solicitações de DNS *unicast*, enquanto (b) implanta uma comunicação *multicast* para propagar solicitações de DNS localmente.

destinos dos pacotes é limitada a apenas três endereços, um servidor do próprio fabricante e servidores DNS e NTP. Por outro lado, a Figura 5.1(b) mostra a comunicação do Xiaomi Hub que, como pode ser observado, é bem diferente do Blink Security Hub. O Xiaomi Hub usa majoritariamente o protocolo UDP na camada de transporte, além de utilizar DNS *multicast* para resolver nomes de *host* de dispositivos locais. Também é possível notar que o Xiaomi Hub emprega o protocolo HTTP, ao invés do HTTPS, como o primeiro dispositivo, e portas da camada de transporte de usuários não-privilegiados, na faixa de de 1024 a 49151.

Ao analisar os pacotes de comunicação dos dispositivos, é possível notar que também há uma diferença entre o intervalo entre pacotes e a duração dos fluxos. Um fluxo combina todos os pacotes que contêm valores idênticos para cinco atributos: protocolo, endereço de origem e destino e porta de origem e destino. A duração do fluxo consiste no intervalo entre o primeiro e o último pacote recebido ou enviado. Através da Figura 5.2, é possível

observar que mais de 80% dos fluxos duram até 10 segundos, enquanto 97% de todos os fluxos possuem até 60 segundos de duração. Dessa forma, para extrair estatísticas relevantes, é necessário realizar uma análise temporal coerente com a duração dos fluxos e adotar uma janela temporal condizente com as características observadas. Assim, 60 segundos é o intervalo de tempo escolhido como a janela de tempo para a análise de tráfego no restante deste trabalho, pois abrange a duração total da maioria dos fluxos presentes no conjunto de dados e permite que fluxos de longa duração sejam bem representados, sem fragmentação excessiva.

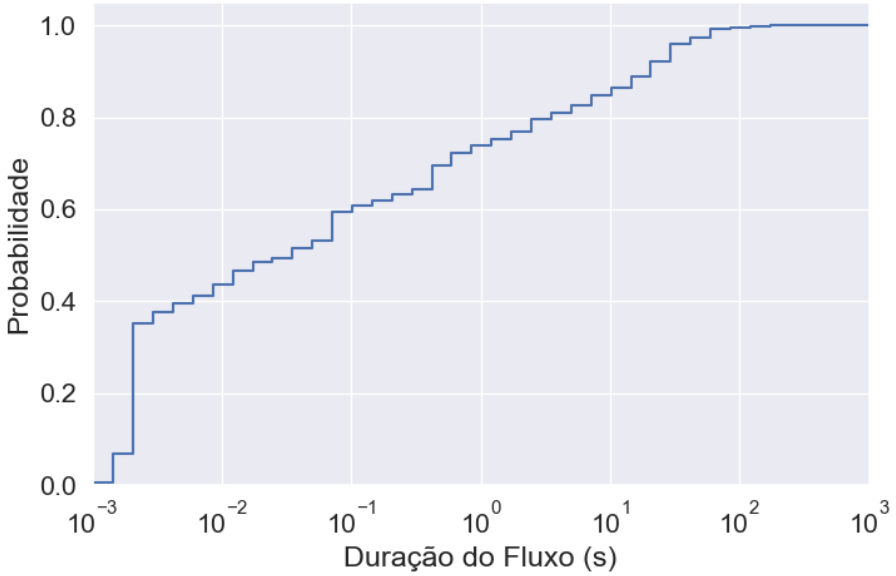


Figura 5.2: CDF para a duração do fluxo. Uma janela de tempo de 60 s abrange até 97% de todos os fluxos no conjunto de dados.

5.1.2 Pré-processamento dos dados

O conjunto de dados consiste em diversos arquivos do tipo `pcap`, contendo dados brutos e irregulares. Assim, uma etapa de pré-processamento é obrigatória para analisá-los e utilizá-los corretamente. Os arquivos `pcap` são processados usando a biblioteca `PyShark`[3] e, então, os campos de interesse são extraídos e salvos em uma estrutura DataFrame da biblioteca `Pandas`[4].

Os pacotes considerados nesta análise utilizam o protocolo IP na camada de rede e UDP ou TCP como protocolo da camada de transporte. Os pacotes da camada de enlace e os pacotes de controle, como o ICMP, foram descartados. Os campos extraídos dos

[3]Disponível em `https://github.com/KimiNewt/pyshark/`.
[4]disponível em `https://pandas.pydata.org/`.

pacotes são portas de origem e destino, endereço IP de origem e destino, endereço MAC de origem e destino, protocolo da camada de transporte, tamanho do pacote, *flags* TCP, tamanho do segmento TCP, carimbo de data/hora, protocolo da camada de aplicação. Para pacotes UDP, os campos referentes ao protocolo TCP foram zerados. O conjunto de dados resultante contém mais de seis milhões de pacotes.

A etapa de pré-processamento não considera a carga útil do pacote, pois os dispositivos foram caracterizados de acordo com seu padrão estatístico de fluxo e não com o conteúdo dos dados, uma vez que essa prática é empregável até mesmo com pacotes que utilizem criptografia dos dados, como HTTPS e TLS.

Os pacotes são agrupados em fluxos, ou seja, pacotes com os mesmos endereços IP de origem e destino, mesmas portas de origem e destino e mesmo protocolo da camada de transporte. Cada pacote recebe um número de identificação de fluxo e um rótulo de direção que pode assumir o valor "*forward*", se o dispositivo enviar o pacote, ou "*backward*", caso contrário.

Uma análise inicial é realizada para avaliar a duração dos fluxos e escolher um período de monitoramento adequado. A análise é realizada através de janelas de tempo, uma vez que esse método permite capturar as principais características de fluxo, sendo também utilizados em ambientes de classificação *online* [45].

5.2 A Representação de Dados Proposta

As vulnerabilidades de privacidade são exploradas usando algoritmos de aprendizado de máquina para identificar e classificar os dispositivos IoT através de suas características de tráfego de rede. A metodologia proposta utiliza informações estatísticas dos fluxos de rede dos dispositivos e, assim, também se aplica ao tráfego de rede criptografado, uma vez que não depende dos dados de carga útil do pacote. Outra vantagem da abordagem estatística é a independência dos números de porta e endereços IP do *host*, fortalecendo o modelo contra alterações de protocolo ou atualizações do servidor das aplicações.

A arquitetura de análise de dados está ilustrada na Figura 5.3. Após a etapa de preprocessamento, os fluxos são divididos em janelas de tempo com base em seu *timestamp* e nos identificadores de fluxo atribuídos aos pacotes na etapa de pré-processamento. A seguir, calculam-se as estatísticas de fluxo para cada janela de tempo que são utilizadas como entrada para os algoritmos de aprendizado de máquina. Adotou-se uma abordagem que contempla quatro tipos de informações: volume de dados, comportamento temporal,

características de vazão e atributos do protocolo TCP. As características dessa última categoria são fixadas com valor zero para fluxos UDP. Para cada fluxo, essas características são calculadas tanto para os pacotes recebidos quanto para os pacotes enviados, sendo em seguida realizada a soma dos dois subconjuntos. Dessa forma, a representação geral do fluxo engloba características de tráfego em ambos os sentidos.

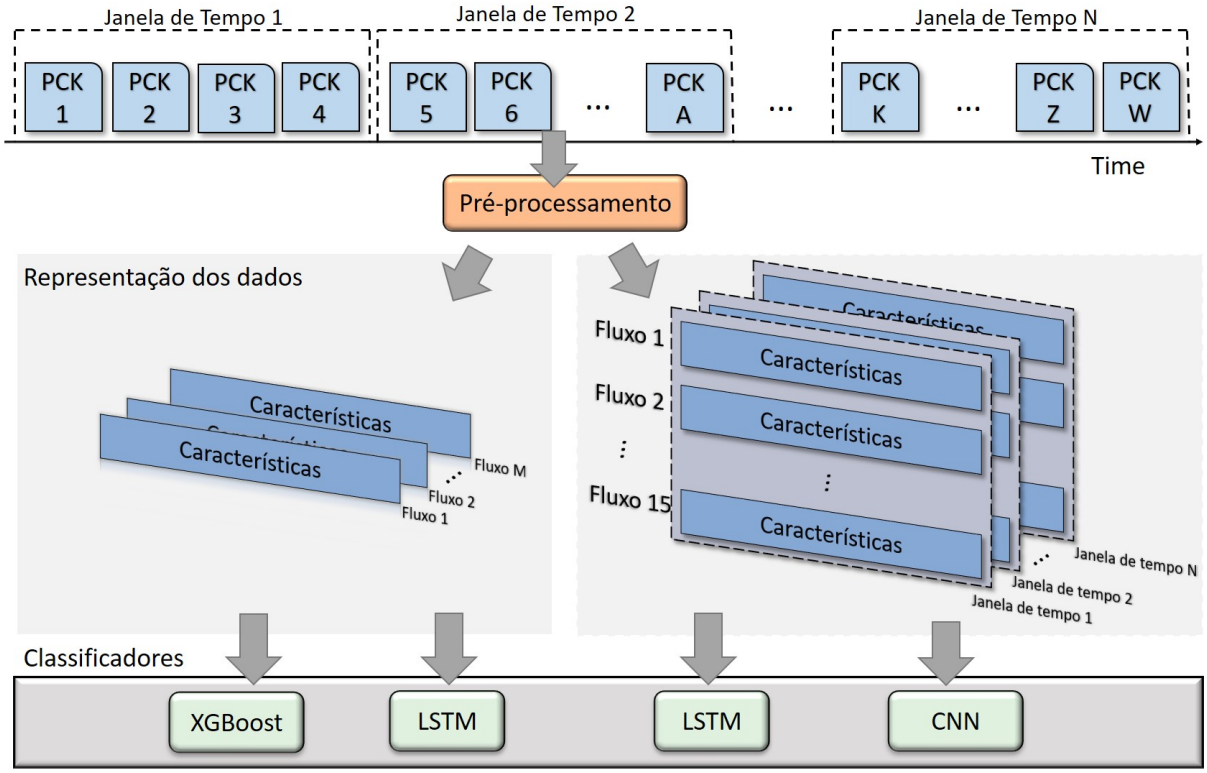


Figura 5.3: A arquitetura proposta para identificação de dispositivos IoT. Os pacotes de rede são divididos em janelas de tempo e agrupados em fluxos. Para cada fluxo, calculam-se recursos estatísticos que alimentam os modelos XGBoost e LSTM-1D. Para cada janela de tempo, as características dos fluxos são agrupadas como uma imagem, que alimenta os modelos CNN e LSTM-2D.

O primeiro conjunto de características é baseado em volume, consistindo na quantidade e tamanho dos pacotes em bytes. Como os pacotes variam em tamanho, calculam-se os valores mínimo, máximo, médio, variância e desvio padrão para representar o volume, além da soma total de bytes para cada fluxo. O segundo grupo de características são temporais, compreendendo a duração do fluxo e o tempo entre chegadas de pacotes, que são representadas com valores estatísticos, assim como as características baseadas em volume. O terceiro grupo de características compreende as informações de vazão, como o número de pacotes por segundo, o número de bytes por segundo e a proporção entre as taxas de recepção e transmissão. Por fim, as características relacionadas ao protocolo TCP correspondem ao número de pacotes, nos quais, cada *flag* no cabeçalho TCP está

ativa e aos valores estatísticos do comprimento do segmento TCP.

O resultado da extração de características é um conjunto de dados agrupados por janelas temporais com 294.000 fluxos de 33 dispositivos agrupados em um tensor de 76 características [5]. Observa-se que algumas características possuem *outliers* significativos. Assim, o valor máximo de cada característica é limitado ao valor do 90^o percentil e o limite inferior ao 10^o percentil. A limitação dos valores é necessária, pois diversos algoritmos de aprendizado de máquina requerem dados normalizados para realizar a classificação e a não-exclusão dos *outliers* pode comprometer significativamente a capacidade de classificação, uma vez que valores muito grandes ou muito pequenos podem tornar valores medianos em valores irrisórios devido a normalização.

5.3 Algoritmos de Classificação

Dentre todos os modelos de aprendizado de máquina disponíveis, foram selecionados alguns modelos conhecidos da literatura que apresentam bom desempenho em tarefas de classificação para realizar a identificação de dispositivos através da análise dos dados de rede. O objetivo de utilizar diferentes algoritmos é verificar a adequação e os pontos fortes e fracos de cada implementação. Os modelos escolhidos são: árvores de decisão XGBoost, CNN e LSTM. Todos os modelos usam as mesmas características de fluxo para a classificação, mas a organização dos dados difere para cada modelo, de forma a corresponder às suas especificações de entrada. A seguir, apresenta-se uma breve descrição de cada algoritmo avaliado com os parâmetros e arquiteturas utilizados, assim como a representação dos dados em cada cenário.

5.3.1 Árvore *XGBoost*

Na abordagem utilizada, cada amostra de entrada do algoritmo é um vetor que corresponde a um fluxo contendo suas 76 características. Na implementação do algoritmo, a classificação da árvore é realizada através da probabilidade de uma entrada pertencer a cada uma das classes. Os dispositivos IoT, que representam as classes a serem classificadas, são representados por números inteiros para compatibilidade com a implementação do algoritmo do pacote `XGBoost`[6] implementado em Python.

[5] O dispositivo Insteon Hub não possui dados de tráfego suficientes para ser analisado e foi excluído da análise.

[6] Disponível em https://xgboost.readthedocs.io.

5.3.2 CNN

A representação de dados para as redes CNN consiste em uma imagem de dimensões $w \times h$, na qual as características são colunas e cada linha representa um fluxo. A largura da imagem w é definida como 76, representando as características estatísticas do fluxo. Para escolher a altura h, é utilizado o número de fluxos ativos na mesma janela de tempo. A Figura 5.4 representa a Função de Distribuição Cumulativa (CDF) dos fluxos ativos, no qual é possível observar que 90% das janelas de tempo contêm no máximo 15 fluxos ativos. Assim, define-se a altura da imagem h como 15. Se uma janela de tempo contiver menos de 15 fluxos, não haverá fluxos suficientes para preencher as linhas da imagem, de modo que as linhas restantes serão preenchidas com zeros. Por outro lado, janelas de tempo com mais de 15 fluxos são ajustadas descartando fluxos aleatoriamente.

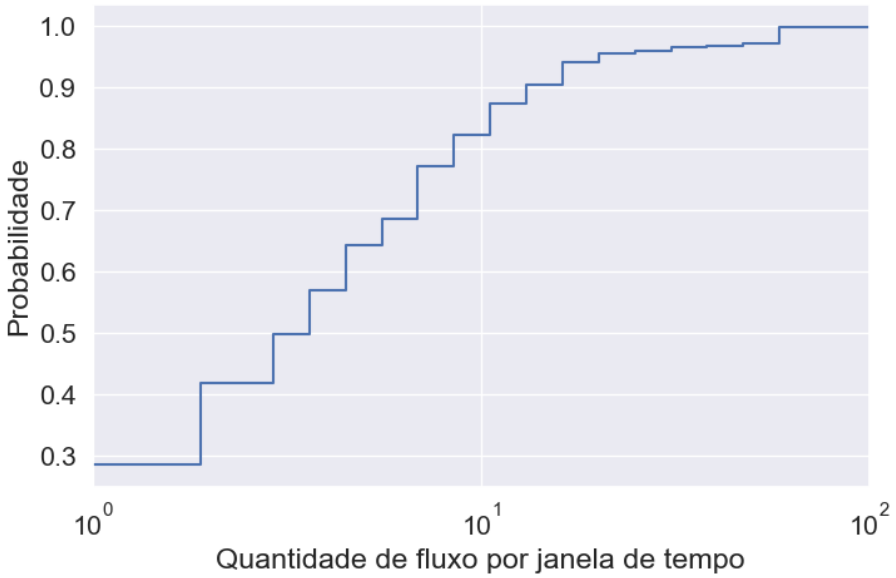


Figura 5.4: CDF do número de fluxo por janela. Uma janela de 60 s atinge 0,9 de probabilidade de conter até 15 fluxos.

O modelo CNN é construído usando a API `Keras` [7] em Python. O formato de entrada desta algoritmo tem quatro dimensões no formato (j, h, w, p), onde j é o tamanho do lote , h a altura , w a largura e p profundidade. Como a representação de dados escolhida não tem a dimensão profundidade e a imagem da janela de tempo tem o formato (h, w, 1), concatenam-se as janelas para obter um tensor no formato (j, h, w, 1), onde j é a quantidade de janelas de tempo, com valor de 44.645 para o conjunto de dados processado.

Implementam-se três modelos de CNN para avaliar o desempenho do algoritmo e escolher a melhor arquitetura para responder ao problema de identificação do dispositivo.

[7] Disponível em https://keras.io/.

Também é realizada uma avaliação do impacto da adição de camadas no resultado geral do classificador em termos de desempenho de classificação, tempo de treinamento e tempo de predição. Com esse objetivo, define-se um bloco básico contendo três camadas. A primeira camada é uma camada convolucional com um *kernel* 3×3 e um filtro de tamanho 32, seguido por uma camada de *Max-Pooling* e uma camada de normalização em lote. O modelo de CNN mais simples implementado possui apenas uma cópia desse bloco básico; o segundo apresenta dois blocos; e o terceiro modelo, três blocos. Todos os três modelos possuem uma camada de achatamento (*flattening*), uma camada totalmente conectada contendo 50 nós, com função de retificação linear, *Rectified Linear Unit* (ReLU), como função de ativação e uma camada de normalização em lote. A última camada é totalmente conectada com 33 nós com uma função de ativação softmax que fornece o resultado da classificação como a probabilidade de cada classe.

5.3.3 LSTM

São utilizadas duas abordagens diferentes para treinar a rede LSTM, conforme ilustrado na Figura 5.3. Na primeira abordagem, a entrada é um fluxo único. Assim, os fluxos são apresentados, processados e classificados individualmente, da mesma forma que é feito no modelo de XGBoost. Na segunda abordagem, a entrada LSTM são os fluxos agrupados em janelas temporais. Portanto, o algoritmo processa e classifica janelas temporais de forma semelhante ao processamento de imagens da CNN. Em ambas as abordagens, o LSTM processa as entradas ordenadas temporalmente para aprender os seus padrões temporais. O modelo LSTM também foi implementado através da API Keras.

5.4 Avaliação e resultados da proposta

Todos os experimentos foram realizados em um computador equipado com CPU Intel i7 9700k @ 3.0 GHz, placa de vídeo NVIDIA GeForce GTX 1660 Super (6 GB) e memória de 32 GB.

Foi utilizada uma abordagem de validação cruzada de dez vezes para o treinamento do modelo, de forma a permitir a avaliação da proposta com diferentes arranjos dos dados e reproduzir o treinamento várias vezes, verificando a relevância estatística do resultado obtido. O algoritmo utilizado para dividir as amostras para a validação cruzada é o StratifiedKFold, disponível na biblioteca `scikit-learn` [8], que divide o conjunto de

[8]Disponível em https://scikit-learn.org/stable/.

amostras em subconjuntos contendo aproximadamente a mesma proporção de amostras de cada classe que o conjunto completo.

Para treinar a CNN e LSTM, configura-se um limite de 100 épocas e uma condição de parada antecipada com paciência de 10 épocas, o que evita o sobreajuste do modelo. O conjunto de validação para essas duas redes neurais contém 10% do conjunto de treinamento e é usado para controlar e acionar o retorno de chamada de parada antecipada.

O desempenho dos modelos é avaliado de acordo com as seguintes métricas: acurácia, precisão, *recall* e F1-score. As métricas são definidas em termos de verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN), falsos negativos (FN), conforme mostrado nas Equações 5.1, 5.2, 5.3 e 5.4. As métricas são calculadas usando o módulo *classification_report* da `scikit-learn`. Por se tratar de uma classificação multiclasse, a precisão e o *recall* são calculados de acordo com cada classe e o valor final, atribuído como resultado do modelo, é uma média dos resultados individuais de cada classe ponderada de acordo com a representatividade da classe no conjunto de dados.

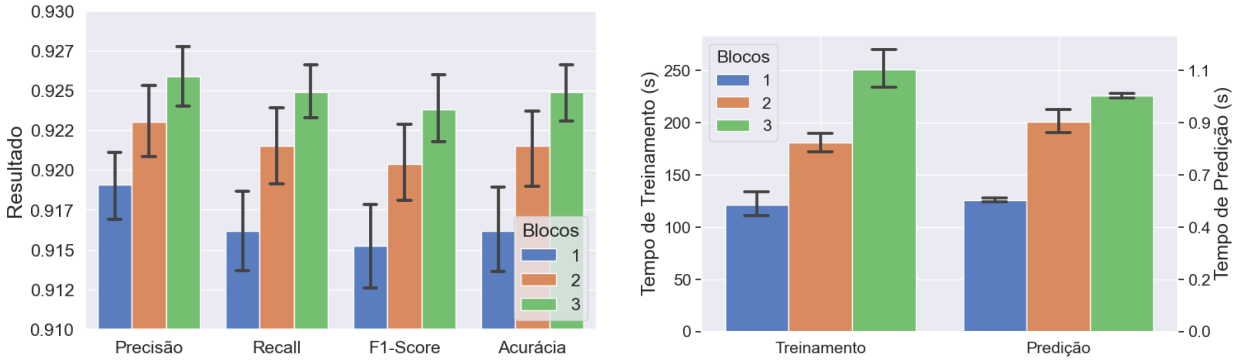
$$Acurácia = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{5.1}$$

$$Precisão = \frac{TP}{(TP + FP)} \tag{5.2}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{5.3}$$

$$F1 - Score = 2 \times \frac{Precisão \times Recall}{(Precisão + Recall)} \tag{5.4}$$

Os modelos também são avaliados em relação aos tempos de processamento, representados pelo tempo de treinamento e pelo tempo de predição. O tempo de treinamento do modelo é o intervalo entre o início do treinamento e o instante onde a condição de parada do treinamento é atingida. Enquanto o tempo de predição é o tempo necessário para classificar todas as amostras no conjunto de teste. Todos os intervalos de tempo são medidos através da execução de apenas um treinamento ou predição por vez.



(a) Desempenho de classificação.

(b) Tempo de treinamento e predição.

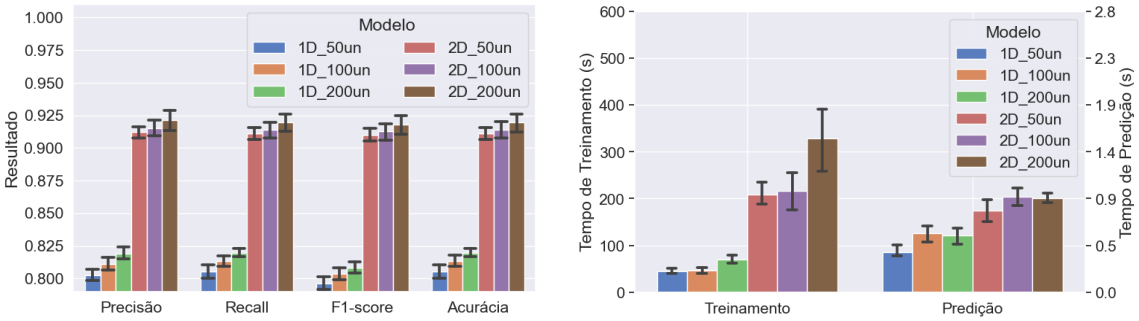
Figura 5.5: Comparação entre os modelos CNN com 1, 2 e 3 camadas de convolução, *max pooling* e normalização de lote. (a) desempenho da classificação, e (b) tempo de treinamento e predição. O modelo CNN com 3 camadas alcança os melhores resultados de classificação com um pequeno aumento no tempo de treinamento e predição.

5.4.1 CNN

A Figura 5.5(a) mostra os resultados para os três modelos CNN analisados. Vale ressaltar que todas as redes tiveram desempenho superior a 90% para classificação de dispositivos em todas as quatro métricas e que o desempenho melhora gradativamente com a adição do bloco de 3 camadas. O tempo de treinamento e predição, conforme visto na Figura 5.5(b), mostra que o crescimento do desempenho se dá ao custo do aumento da complexidade da rede e, consequentemente com o aumento do tempo de processamento. Embora adicionar complexidade à arquitetura de rede melhore o desempenho da classificação, o tempo de processamento adicional pode ser prejudicial para aplicativos que trabalham com uma grande quantidade de dados, como aplicativos de análise de fluxo de rede em tempo real. Dessa forma, a escolha entre os três diferentes modelos deve levar em conta os dois fatores, o desempenho de classificação e os tempos gastos para treinamento e predição.

5.4.2 LSTM

O modelo de rede LSTM é avaliado seguindo duas abordagens diferentes. A primeira abordagem utiliza uma entrada bidimensional, que é semelhante ao conceito de imagem usado na rede CNN. A segunda abordagem usa uma entrada unidimensional, representando um único fluxo com 76 características, semelhantes ao modelo XGBoost. Ambos os modelos contêm uma camada LSTM e uma camada totalmente conectada com 33 nós com a uma função de ativação softmax. A saída de classificação do modelo LSTM é a



(a) Desempenho de classificação.

(b) Tempo de treinamento e predição.

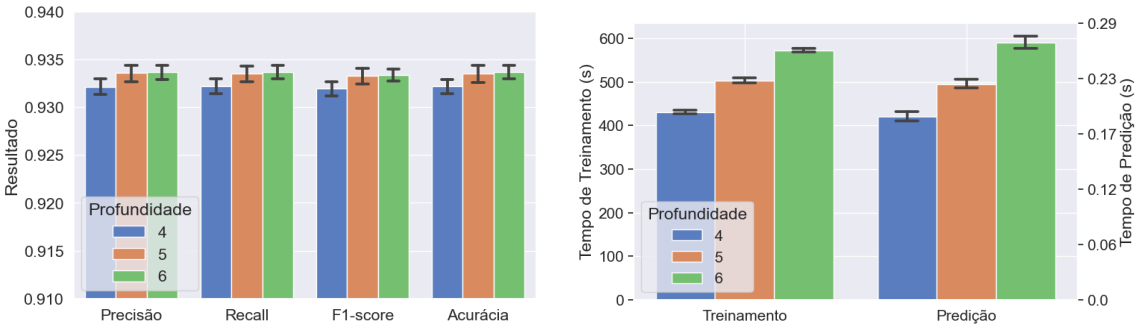
Figura 5.6: Comparação entre os modelos LSTM com entradas 1D e 2D, com 50, 100 e 200 unidades: (a) desempenho da classificação, e (b) tempo de treinamento e predição. O modelo LSTM com entrada 2D e 200 unidades alcança os melhores resultados de classificação com um aumento no tempo de treinamento.

probabilidade do fluxo pertencer a cada classe. Em ambas as abordagens, são implementados modelos com diferentes número de unidades na camada LSTM e o desempenho de cada uma das configurações é analisado, buscando escolher a melhor configuração.

A Figura 5.6(a) mostra o resultado dos modelos LSTM testados. O modelo de entrada bidimensional tem um desempenho até 10% melhor em todas as métricas analisadas do que o modelo LSTM de entrada unidimensional. Esse resultado demonstra que, para redes LSTM, agrupar fluxos em janelas de tempo otimiza o desempenho do modelo, o que pode ser explicado pela maior quantidade de informações que um grupo de fluxos representa quando comparado a um único fluxo. Em relação ao número de unidades, é possível ver uma melhora discreta a medida que os modelos aumentam a quantidade de unidades. Com relação ao tempo de processamento, a Figura 5.6(b) mostra que o tempo de treinamento da entrada bidimensional LSTM é pelo menos quatro vezes maior que o tempo de treinamento da entrada unidimensional LSTM. Assim, é possível concluir que, novamente, a escolha entre os modelos é um compromisso entre tempo de processamento e desempenho.

5.4.3 XGBoost

O modelo XGBoost implementa a abordagem de classificação de fluxo individual e, assim, a entrada do modelo é composta por um vetor de 76 características representando um fluxo. o algoritmo XGBoost é implementado com três valores diferentes para o limite máximo de profundidade para as árvores de decisão: 4, 5 e 6. Encontrar o melhor valor para esse parâmetro é essencial, pois árvores mais profundas podem melhorar o



(a) Desempenho de classificação.

(b) Tempo de treinamento e predição.

Figura 5.7: Comparação entre os modelos XGBoost com valores de profundidade máxima de 4, 5 e 6: (a) desempenho da classificação e (b) tempo de treinamento e predição. Os modelos com valores de profundidade máxima de 5 e 6 alcançam resultados de classificação comparáveis, mas o primeiro é menos complexo e tem menor tempo de treinamento e predição do que o segundo.

desempenho da classificação ao custo de tornar o modelo mais propenso a sobreajuste. A métrica de avaliação utilizada é o erro logístico multiclasse (*multiclass logloss*) e os demais parâmetros permanecem com seus valores padrão.

A Figura 5.7(a) mostra os resultados obtidos com as diferentes configurações. É possível observar que o modelo utilizando 5 e 6 níveis como tamanho máximo de árvore alcança desempenho semelhante, enquanto o modelo com profundidade máxima de 4 níveis apresenta desempenho um pouco inferior. Assim, é possível perceber que um aumento na complexidade do modelo é capaz de melhorar o desempenho do modelo, mas de forma limitada. A comparação do tempo de treinamento e predição é ilustrada na Figura 5.7(b). Conforme esperado, o aumento da complexidade reflete no aumento gradual dos tempos para processar o modelo de 4, 5 e 6 níveis.

5.4.4 Dispositivos Semelhantes Agrupados

Todos os modelos pontuaram acima de 90% em todas as métricas avaliadas e tiveram desempenho semelhante para a maioria dos dispositivos. A Figura 5.8 mostra o resultado da classificação do modelo CNN com três camadas de convolução, detalhado por dispositivo. É possível observar que os dispositivos que tiveram os piores resultados de classificação foram os assistentes domésticos: *Google Home*, *Google Home Mini*, *Echodot*, *Echoplus* e *Echospot*.

Ao analisar os resultados através da matriz de confusão, resumida na Figura 5.9, é

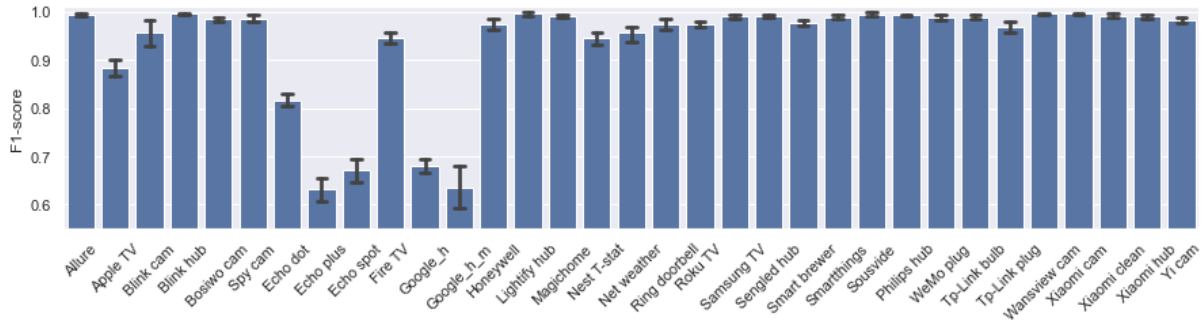


Figura 5.8: F1-Score de cada classe de dispositivo para o modelo de classificação CNN. Assistentes de casa inteligente, como Google Home (`Google_h`), Google Home Mini (`Google_h_m`), Echo Dot e Echo Plus, são as classes que representam o menor desempenho para todos modelos de classificação.

possível observar que o classificador confunde os dispositivos *Google Home* e *Google Home Mini* entre si, assim como os dispositivos *Echodot*, *Echoplus*, *Echospot* são confundidos entre si. Esse resultado pode ser explicado pela semelhança dos dispositivos que possuem os mesmos fabricantes e implementam serviços semelhantes, mudando basicamente os recursos de interface do usuário e poucas funcionalidades. Por simplicidade, não são apresentados os resultados detalhados de cada modelo, mas essa é uma característica observada nos três tipos de algoritmos utilizados na classificação. Dessa forma, levando em consideração a similaridade dos dispositivos, todos os testes são refeitos, agrupando o *Google Home* e o *Google Home Mini* em uma nova classe chamada *Google Home Devices* e os dispositivos *Echodot*, *Echoplus*, *Echospot* na classe *Echo Devices*. Os resultados obtidos são apresentados em seguida, na subseção 5.4.5.

5.4.5 Comparação entre XGBoost, CNN e LSTM

O agrupamento dos dispositivos do tipo assistente doméstico realizado melhorou a capacidade de classificação de todos os modelos de classificadores. Para avaliar os diferentes algoritmos de classificação, são comparados os modelos XGBoost, CNN e LSTM que obtiveram o melhor desempenho na etapa anterior, são eles: (i) CNN com três camadas de convolução, max-*pooling* e normalização em lote; (ii) rede LSTM com 200 unidades e entrada em duas dimensões; e (iii) XGBoost com limite de profundidade igual a 5.

A Figura 5.10 mostra os resultados obtidos com os três modelos. É possível notar que os resultados obtidos com o XGBoost são ligeiramente superiores aos resultados da CNN, alcançando resultado similar, considerando a variação das medidas. Por outro lado, o desempenho do modelo LSTM foi consideravelmente inferior. Um outro ponto importante

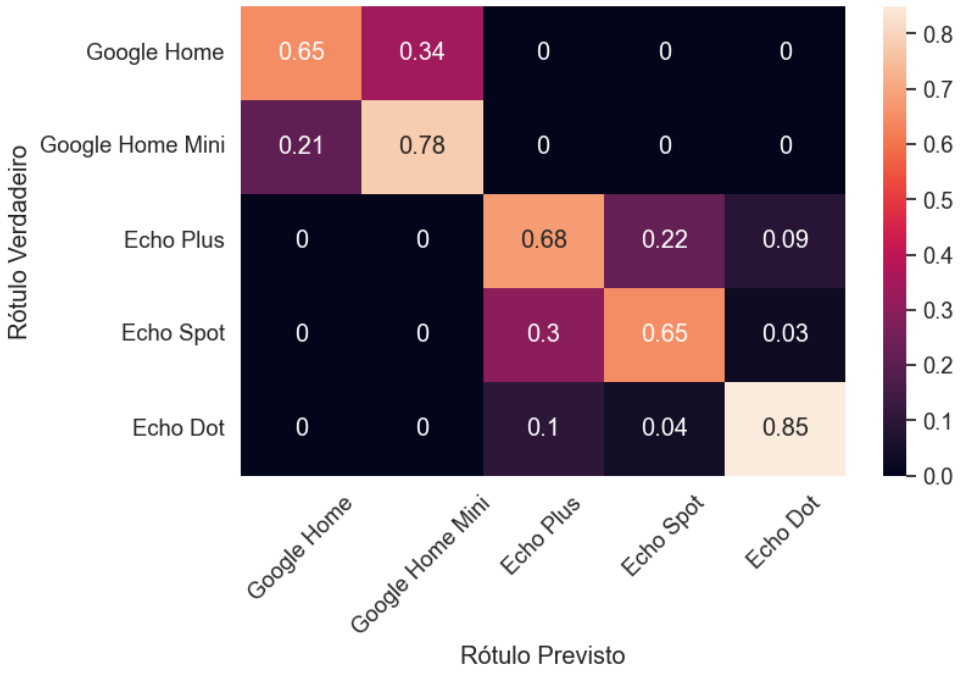
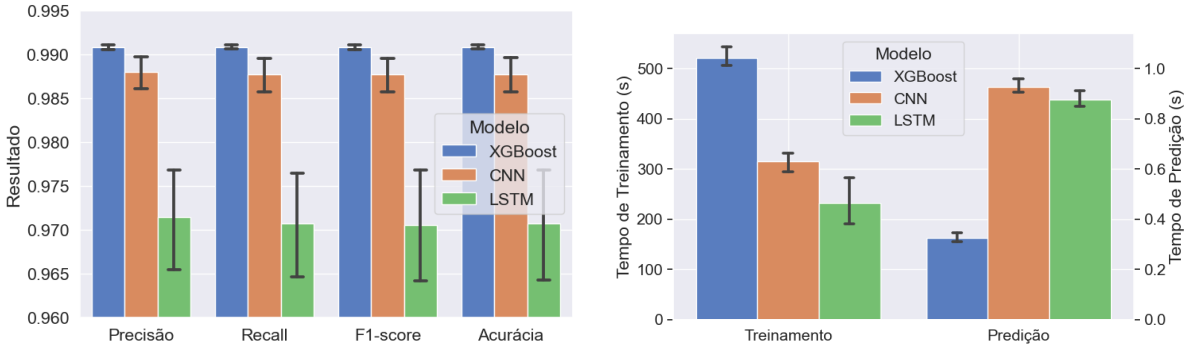


Figura 5.9: Matriz de confusão dos cinco principais dispositivos com as menores taxas de acerto. Os dispositivos do *Home Assistant* apresentam classificações incorretas devido a diferentes versões de cada dispositivo.

para o modelo XGBoost é que esse algoritmo é capaz de realizar uma classificação mais refinada a nível de fluxo, uma vez que a entrada do modelo são fluxos individuais. Já o modelo CNN, apesar de desempenho semelhante, necessita de um conjunto de fluxos dispostos em janelas temporais para realizar a classificação. Por sua vez, o modelo LSTM obteve um resultado aquém dos outros classificadores, indicando que a sua habilidade de aprender as relações temporais não superou a capacidade de classificação dos outros modelos.

Em relação à desvantagem do XGBoost, é possível observar na Figura 5.10(b) no seu alto tempo de treinamento, cujo valor médio é maior que o dobro do tempo de treinamento de uma rede LSTM e consideravelmente superior à CNN. Porém, ao comparar o tempo de predição, o XGBoost é quase três vezes mais rápido que a CNN ou LSTM. Isso pode ser explicado devido ao menor número de operações aritméticas que o XGBoost executa em relação aos cálculos necessários nas abordagens de rede neural.

Através de todos os testes realizados, é possível concluir que os três algoritmos de classificação implementados foram capazes de classificar os dispositivos IoT através da representação estatística dos respectivos tráfegos de rede. No geral, o modelo XGBoost foi capaz de realizar uma classificação mais refinada e com menor tempo de predição.



(a) Desempenho de classificação.

(b) Tempo de treinamento e predição.

Figura 5.10: Comparação entre os modelos XGBoost, CNN e LSTM com 30 classes de dispositivos. (a) desempenho da classificação e (b) tempo de treinamento e predição. O modelo XGBoost alcança os melhores resultados de classificação e o menor tempo de predição, embora tenha o maior tempo de treinamento.

5.5 Conclusão sobre o reconhecimento de dispositivos IoT

Nesta dissertação, foi analisada a capacidade de identificação de dispositivos de Internet das Coisas usando um conjunto de dados de rede real. É proposta uma representação dos dados de tráfego de rede, utilizando uma janela de tempo curta para extrair estatísticas de fluxos, o que permite operações *online*. A representação não utiliza a carga útil do tráfego, nem os endereços contactados, o que torna a abordagem mais robusta a eventuais atualizações de servidores e de mensagens trocadas. A proposta foi testada através de três modelos de aprendizado de máquina, CNN, LSTM e XGBoost, com um conjunto de dados contendo 33 dispositivos. Os resultados mostram que o método proposto atinge mais de 97% de acurácia, precisão e *recall* na identificação dos dispositivos em todos os modelos testados e 99% para a melhor configuração baseada em árvore.

A alta precisão na identificação dos dispositivos ressalta o quanto da privacidade do usuário está sendo violada. Através do trabalho proposto, é possível saber não só o tipo de equipamento usado, mas também qual modelo. Assim, um atacante pode explorar informações baseadas nas categorias dos equipamentos, como acessar as imagens de câmeras ou controlar trancas e dispositivos de automação, como também aproveitar brechas de segurança conhecidas de determinados modelos para realizar ataques contra os usuários. Dessa forma, é importante que os usuários se conscientizem das vulnerabilidades existentes e optem, quando possível, por dispositivos que implementem soluções de segurança e cujos fabricantes sejam transparentes sobre os dados coletados sobre os usuários.

Capítulo 6

Conclusões

Atualmente, há uma crescente preocupação com o consumo consciente, redução de desperdício, assim como diminuição do impacto no ambiente e com gastos de energia. Nesse contexto, há demanda para construir sistemas de comunicações mais versáteis, evitando a ociosidade de recursos e gastos com equipamentos. Da mesma forma, a crescente conectividade de dispositivos ubíquos traz novas preocupações com privacidade das informações dos usuários. Esta dissertação aborda o emprego de dispositivos SDR em ambientes virtualizados e a questão de privacidade de usuário no uso de dispositivos IoT.

Na primeira parte desta dissertação, foram avaliados os limites experimentais dos dispositivos SDRs e o impacto da virtualização nas suas aplicações de comunicações. Os experimentos de rede avaliaram a vazão alcançada e o respectivo uso de CPU em um Linux nativo e em ambientes virtualizados alterando diversos parâmetros e utilizando dois dispositivos SDR USRP. Os resultados obtidos mostram que o desempenho das aplicações SDR executadas em contêineres é muito próximo ao obtido utilizando um Linux nativo. Por outro lado, as aplicações testadas em máquinas virtuais apresentam mais erros e mais perdas de pacotes, além de uma menor vazão de rede. Isso se deve a forma que as máquinas virtuais são implementadas, exigindo um *guest* OS e, por consequência, demandando mais processamento. Devido à implementação em software das aplicações SDR, observou-se que sobrecargas de CPU comprometem diretamente a qualidade das comunicações.

A taxa de transmissão e recepção dos dispositivo USRP foi testada em cada um dos ambientes em três diferentes cenários de comunicação: transmissão *simplex*, recepção *simplex* e *full-duplex*. Os resultados mostram que, nos modos *simplex*, todos os ambientes possuem comportamentos equivalentes em termos de taxa de transmissão, porém a máquina virtual chega a utilizar mais que o dobro de CPU quando comparado ao consumo dos outros ambientes. No modo *full-duplex*, o desempenho do Linux nativo e do contêiner

continuam semelhantes, enquanto a máquina virtual apresenta uma piora no desempenho e um consumo ainda maior de uso de CPU. Também observou-se que o dispositivo USRP N200 apresentou mais erros que o USRP B200, mesmo quando executado em Linux nativo em taxas dentro da especificação do dispositivo, indicando limitações de sua própria arquitetura.

Devido à importância das comunicações OFDM nas comunicações das redes móveis celulares, também foram realizados testes com comunicação OFDM nos dois dispositivos USRP, utilizando diferentes tamanhos de pacotes e números de portadoras. Os resultados mostram que os dois dispositivos se comportam de maneira diferente quanto à variação de cada parâmetro, mas o dispositivo B200 se comunica com a menor perda de dados. Com relação ao desempenho em ambientes virtualizados, não foi observada uma degradação significativa da comunicação devido à virtualização, exceto no cenário em que a aplicação é executada em uma VM com o dispositivo B200.

Os resultados dos experimentos demonstram que é viável o uso de máquinas virtuais e contêiner para o uso de SDR, porém a virtualização de ambientes através de máquinas virtuais apresenta uma maior restrição nas taxas de amostragem utilizadas, obtendo uma grande sobrecarga de processamento para taxas mais altas. O contêiner obteve um desempenho próximo ao Linux Nativo em todos os cenários, demonstrando ser adequado o seu uso nas aplicações SDR, inclusive na virtualização das redes de acesso.

Como trabalho futuro nesse domínio, é previsto avaliar o comportamento do SDR atuando como UE e gNB em um ambiente virtualizado, através da implementação de uma rede 5G privada, utilizando soluções como o OAI. Contempla-se avaliar o uso de diversas instâncias em uma mesma máquina física e verificar o impacto que a execução simultânea causa no desempenho geral do sistema.

Na segunda parte desta dissertação, é proposto um método para identificar dispositivos IoT através da análise do tráfego de rede, usando técnicas de aprendizado de máquina baseadas em características de fluxo de rede, englobando também o tráfego de rede criptografado. O trabalho proposto analisa apenas dados estatísticos de tráfego da rede para reconhecer os dispositivos IoT na rede e, assim, revelar o comportamento e as preferências dos usuários. Além disso, a proposta emprega uma janela de tempo de apenas 60 segundos para extrair estatísticas de fluxos, o que permite operações online. A proposta foi avaliada usando três modelos de aprendizado de máquina com um conjunto de dados contendo 33 dispositivos. Os resultados mostram que o método proposto pode atingir 99% de precisão, acurácia e *recall* na melhor configuração do modelo baseada em árvore

de decisão.

Como trabalho futuro, pretende-se explorar a identificação do modo de operação dos dispositivos IoT que tenham o potencial de fornecer informações ainda mais significativas sobre o comportamento e o perfil dos usuários. Por exemplo, o tráfego de rede de uma câmera ou uma lâmpada inteligente ativada por movimento, pode indicar que alguém está em casa e a análise diária dos dados pode revelar os hábitos do usuário. Também vislumbra-se como trabalho futuro prever um mecanismo de proteção contra ataques orquestrados contra dispositivos IoT.

Esta dissertação reúne o conteúdo de artigos publicados nos congressos *Conference on Innovation in Clouds, Internet and Networks* (ICIN-2022) [46], *Cyber Security in Networking Conference* (CSNet-2022) [47], *Workshop* em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance-2022) [48] e em um minicurso no Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg-2021) [49].

Referências

[1] GARTNER Forecasts Worldwide 5G Network Infrastructure Revenue to Grow 39% in 2021. Agosto 2021. Acessado em 10 de abril de 2023. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2021-08-04-gartner-forecasts-worldwide-5g-network-infrastrucutre-revenue-to-grow-39pc-in-2021>.

[2] DAWS, R. *Kaspersky: Attacks on IoT devices double in a year*. Setembro 2021. Acessado em 10 de abril de 2023. Disponível em: <https://iottechnews.com/news/2021/sep/07/kaspersky-attacks-on-iot-devices-double-in-a-year/>.

[3] MATTOS, D. M. F.; VELLOSO, P. B.; DUARTE, O. C. M. B. An agile and effective network function virtualization infrastructure for the internet of things. *Journal of Internet Services and Applications*, v. 10, n. 1, p. 6, Mar 2019. ISSN 1869-0238.

[4] SANZ, I. J.; MATTOS, D. M. F.; DUARTE, O. C. M. B. SFCPerf: An automatic performance evaluation framework for service function chaining. In: *Proceedings of the 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS)*. [S.l.: s.n.], 2018. p. 1–9.

[5] O que é virtualização? Amazon, 2003. Acessado em 10 de abril de 2023. Disponível em: <https://aws.amazon.com/pt/what-is/virtualization/#:~:text=Virtualiza%C3%A7%C3%A3o%20%C3%A9%20a%20tecnologia%20que,em%20uma%20%C3%BAnica%20m%C3%A1quina%20f%C3%ADsica.>.

[6] INTRODUCING 3GPP. Acessado em 10 de abril de 2023. Disponível em: <https://www.3gpp.org/about-us/introducing-3gpp>.

[7] VIEW on 5G Architecture Version 3.0. [S.l.], Fevereiro 2020. 182 p. Disponível em: <http://doi.org/10.5281/zenodo.3265031>.

[8] KIST, M.; ROCHOL, J.; DASILVA, L. A.; BOTH, C. B. SDR virtualization in future mobile networks: Enabling multi-programmable air-interfaces. In: IEEE. *Proceedings of the IEEE International Conference on Communications (ICC)*. Kansas City, MO, USA, 2018. p. 1–6.

[9] 5G System Overview. 3GPP, Aug 2022. Acessado em 10 de abril de 2023. Disponível em: <https://www.3gpp.org/technologies/5g-system-overview>.

[10] CHECKO, A.; CHRISTIANSEN, H. L.; YAN, Y.; SCOLARI, L.; KARDARAS, G.; BERGER, M. S.; DITTMANN, L. Cloud RAN for mobile networks—a technology overview. *IEEE Communications Surveys Tutorials*, v. 17, n. 1, p. 405–426, 2015.

[11] LARSEN, L. M.; CHECKO, A.; CHRISTIANSEN, H. L. A survey of the functional splits proposed for 5G mobile crosshaul networks. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 1, p. 146–172, 2018.

[12] GARCIA-SAAVEDRA, A.; COSTA-PéREZ, X. O-RAN: Disrupting the virtualized RAN ecosystem. *IEEE Communications Standards Magazine*, v. 5, n. 4, p. 96–103, 2021.

[13] STUDY on New Radio Access Technology: Radio Access Architecture and Interface (Release 14). [S.l.], Março 2017.

[14] DIAS, W.; FERREIRA, A.; KAGAMI, R.; FERREIRA, J. S.; SILVA, D.; MENDES, L. 5G-RANGE: A transceiver for remote areas based on software-defined radio. In: IEEE. *Proceedings of the 2020 European Conference on Networks and Communications*. Dubrovnik, Croatia, 2020. p. 100–104.

[15] MITOLA, J. Software radios: Survey, critical evaluation and future directions. *IEEE Aerospace and Electronic Systems Magazine*, IEEE, v. 8, n. 4, p. 25–36, 1993.

[16] RESEARCH, E. *USRP Hardware Driver and USRP Manual*. [S.l.]. Disponível em: <`https://files.ettus.com/manual/index.html`>.

[17] SCHMID, T.; SEKKAT, O.; SRIVASTAVA, M. B. An experimental study of network performance impact of increased latency in software defined radios. In: *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*. New York, NY, USA: Association for Computing Machinery, 2007. (WinTECH '07), p. 59–66. ISBN 9781595937384.

[18] BECKER, J. K.; GVOZDENOVIC, S.; XIN, L.; STAROBINSKI, D. Testing and fingerprinting the physical layer of wireless cards with software-defined radios. *Computer Communications*, Elsevier, v. 160, p. 186–196, 2020.

[19] CHANG, H. C.; QIU, B. J.; CHIU, C. H.; CHEN, J. C.; LIN, F. J.; BASTIDA, D. D. L.; LIN, B. S. P. Performance evaluation of Open5GCore over KVM and Docker by using Open5GMTC. In: IEEE. *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan, 2018. p. 1–6.

[20] MATTOS, D. M.; FERRAZ, L. H. G.; COSTA, L. H. M.; DUARTE, O. C. M. Evaluating virtual router performance for a pluralist future internet. In: *Proceedings of the 3rd international conference on information and communication systems*. Irbid, Jordan: [s.n.], 2012. p. 1–7.

[21] EIRAS, R. S.; COUTO, R. S.; RUBINSTEIN, M. G. Performance evaluation of a virtualized http proxy in kvm and docker. In: IEEE. *Proceedings of the 7th International Conference on the Network of the Future (NOF)*. [S.l.], 2016. p. 1–5.

[22] MORABITO, R.; KJÄLLMAN, J.; KOMU, M. Hypervisors vs. lightweight virtualization: a performance comparison. In: IEEE. *Proceedings of the 2015 IEEE International Conference on cloud engineering*. [S.l.], 2015. p. 386–393.

[23] BACHIEGA, N. G.; SOUZA, P. S.; BRUSCHI, S. M.; SOUZA, S. D. R. D. Container-based performance evaluation: A survey and challenges. In: IEEE. *Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E)*. [S.l.], 2018. p. 398–403.

[24] BLOESSL, B.; SEGATA, M.; SOMMER, C.; DRESSLER, F. Towards an open source IEEE 802.11 p stack: A full sdr-based transceiver in GNU Radio. In: IEEE. *Proceedings of the 2013 IEEE Vehicular Networking Conference*. Boston, MA, USA, 2013. p. 143–149.

[25] DZOGOVIC, B.; FENG, B.; DO, T. V. Building virtualized 5g networks using open source software. In: IEEE. *Proceedings of the 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. Penang, Malaysia, 2018. p. 360–366.

[26] LAI, W.-P.; WANG, Y.-H.; CHIU, K.-C. Containerized design and realization of network functions virtualiza-tion for a light-weight evolved packet core using openairinterface. In: IEEE. *Proceedings of the 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. [S.l.], 2018. p. 472–477.

[27] VMWARE. *White paper: Network Throughput in a Virtual Infrastructure*. [S.l.], 2005. 10 p. Disponível em: <`https://www.vmware.com/pdf/esx_network_planning.pdf`>.

[28] VIEW on 5G Architecture Version 4.0. [S.l.], Outubro 2021. 223 p. Disponível em: <`https://doi.org/10.5281/zenodo.5155657`>.

[29] BALDINI, G.; STURMAN, T.; BISWAS, A. R.; LESCHHORN, R.; GODOR, G.; STREET, M. Security aspects in software defined radio and cognitive radio networks: A survey and a way ahead. *IEEE Communications Surveys & Tutorials*, IEEE, v. 14, n. 2, p. 355–379, 2011.

[30] LI, K.; YU, X.; ZHANG, H.; WU, L.; DU, X.; RATAZZI, P.; GUIZANI, M. Security mechanisms to defend against new attacks on software-defined radio. In: IEEE. *Proceedings of international conference on computing, networking and communications (ICNC)*. Maui, HI, USA, 2018. p. 537–541.

[31] HILL, R.; MYAGMAR, S.; CAMPBELL, R. Threat analysis of GNU software radio. In: CITESEER. *Proceedings of the World Wireless Congress (WWC 2005), Palo Alto, California, USA (24–27 May 2005)*. [S.l.], 2005.

[32] SIERRA-ARRIAGA, F.; BRANCO, R.; LEE, B. Security issues and challenges for virtualization technologies. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 53, n. 2, p. 1–37, 2020.

[33] DESIKAN, T. *Docker hub security vulnerabilities*. Maio 2015. Aessado em 10 de abril de 2023. Disponível em: <`https://www.banyansecurity.io/blog/over-30-of-official-images-in-docker-hub-contain-high-priority-security-vulnerabilities`>.

[34] SHU, R.; GU, X.; ENCK, W. A study of security vulnerabilities on docker hub. In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. [S.l.: s.n.], 2017. p. 269–280.

[35] DUTTA, A.; HAMMAD, E. 5g security challenges and opportunities: a system approach. In: IEEE. *Proceedings of the 2020 IEEE 3rd 5G World Forum (5GWF)*. [S.l.], 2020. p. 109–114.

[36] ACAR, A.; FEREIDOONI, H.; ABERA, T.; SIKDER, A. K.; MIETTINEN, M.; AKSU, H.; CONTI, M.; SADEGHI, A.-R.; ULUAGAC, S. Peek-a-boo: I see your smart home activities, even encrypted! In: *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. [S.l.: s.n.], 2020. p. 207–218.

[37] LI, J.; LI, Z.; TYSON, G.; XIE, G. Your privilege gives your privacy away: An analysis of a home security camera service. In: IEEE. *Proceedings of the 2020 IEEE Conference on Computer Communications Workshops (INFOCOM)*. [S.l.], 2020. p. 387–396.

[38] REN, J.; DUBOIS, D. J.; CHOFFNES, D.; MANDALARI, A. M.; KOLCUN, R.; HADDADI, H. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In: *Proceedings of the 2019 Internet Measurement Conference*. [S.l.: s.n.], 2019. p. 267–279.

[39] CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 785–794.

[40] SIVANATHAN, A.; GHARAKHEILI, H. H.; LOI, F.; RADFORD, A.; WIJENAYAKE, C.; VISHWANATH, A.; SIVARAMAN, V. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, IEEE, v. 18, n. 8, p. 1745–1759, 2018.

[41] SHAHID, M. R.; BLANC, G.; ZHANG, Z.; DEBAR, H. Iot devices recognition through network traffic analysis. In: IEEE. *Proceedings of the 2018 IEEE international conference on big data*. [S.l.], 2018. p. 5187–5192.

[42] MEIDAN, Y.; BOHADANA, M.; SHABTAI, A.; GUARNIZO, J. D.; OCHOA, M.; TIPPENHAUER, N. O.; ELOVICI, Y. Profiliot: A machine learning approach for iot device identification based on network traffic analysis. In: *Proceedings of the symposium on applied computing*. [S.l.: s.n.], 2017. p. 506–509.

[43] BAI, L.; YAO, L.; KANHERE, S. S.; WANG, X.; YANG, Z. Automatic device classification from network traffic streams of internet of things. In: IEEE. *Proceedings of the 2018 IEEE 43rd conference on local computer networks (LCN)*. [S.l.], 2018. p. 1–9.

[44] LOPEZ-MARTIN, M.; CARRO, B.; SANCHEZ-ESGUEVILLAS, A.; LLORET, J. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, IEEE, v. 5, p. 18042–18050, 2017.

[45] LOPEZ, M. A.; MATTOS, D. M. F.; DUARTE, O. C. M. B.; PUJOLLE, G. A fast unsupervised preprocessing method for network monitoring. *Annals of Telecommunications*, v. 74, n. 3, p. 139–155, Apr 2019. ISSN 1958-9395.

[46] BEZERRA, G. M.; FERREIRA, T.; MATTOS, D. M. A precise flow representation for autonomous iot-devices reconnaissance. In: IEEE. *Proceedings of the 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*. [S.l.], 2022. p. 56–63.

[47] BEZERRA, G. M. G.; FERREIRA, T. N.; MATTOS, D. M. Assessing software-defined radio security and performance in virtualized environments for cloud radio

access networks. In: IEEE. *Proceedings of the 6th Cyber Security in Networking Conference (CSNet)*. [S.l.], 2022. p. 1–7.

[48] BEZERRA, G. M. G.; FERREIRA, T. N.; MATTOS, D. M. Uma avaliação de desempenho da implantação de rádio definido por software em ambientes virtualizados para redes de acesso sem fio em nuvem. In: SBC. *Anais do XXI Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. [S.l.], 2022. p. 72–83.

[49] BARBOSA, G. N.; BEZERRA, G. M. G.; MEDEIROS, D. S. de; LOPEZ, M. A.; MATTOS, D. M. Segurança em redes 5g: Oportunidades e desafios em detecção de anomalias e predição de tráfego baseadas em aprendizado de máquina. *Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, 2021.