MINICURSOS

44º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

44º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

Praia do Forte, BA, 25 a 29 de maio de 2026

MINICURSOS

Editora
Sociedade Brasileira de Computação – SBC

Organizadores
Leobino Nascimento Sampaio, UFBA
Allan Edgard Silva Freitas, IFBA
Anelise Munaretto, UTFPR
Igor Monteiro Moraes, UFF
Rodrigo Brandão Mansilha, UNIPAMPA

Realização
Universidade Federal da Bahia – UFBA
Instituto Federal da Bahia – IFBA
Sociedade Brasileira de Computação – SBC
Laboratório Nacional de Redes de Computadores – LARC

Capa: Rodrigo Mansilha

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (44. : 25 – 29 maio 2026 : Praia do Forte, BA)

Minicursos do SBRC 2026 [recurso eletrônico] / organização: Leobino Nascimento Sampaio ... [et al.]. – Dados eletrônicos. – Porto Alegre : Sociedade Brasileira de Computação, 2026.

256 p. : il. : PDF.

Modo de acesso: World Wide Web.
Inclui bibliografia
ISBN 978-85-7669-669-8 (e-book)

1. Computação – Brasil – Evento. 2. Redes de computadores. 3. Sistemas computacionais. 4. Sistemas distribuídos. I. Sampaio, Leobino Nascimento. II. Freitas, Allan Edgar Silva. III. Munaretto, Anelise. IV. Moraes, Igor Monteiro. V. Mansilha, Rodrigo Brandão. VI. Sociedade Brasileira de Computação. VII. 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) 992526018
sbc@sbc.org.br

Realização

Organização

Patrocínio

Diamante

Ouro

Prata

Bronze

Apoio

Índice

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
`https://www.sbc.org.br`

Laboratório Nacional de Redes de Computadores – LARC

Contato
Universidade Federal do Pará, Instituto de Tecnologia
Fac. Engenharia da Computação e Telecomunicações, Rua Augusto Corrêa, 01
Guamá
66075110 - Belém, PA - Brasil
Telefone: (91) 32017000

CNPJ: 00.580.126/0001-82
`https://www.larc.org.br`

Comissão Especial de Redes de Computadores e Sistemas Distribuídos – CE-RESD

Coordenação
Everton Cavalcante (UFRN)

Comitê Gestor
Atslands Rocha (UFC)
Flávia Delicato (UFF)
Célio Albuquerque (UFF)
Magnos Martinello (UFES)

Mensagem dos Coordenadores de Minicursos

Este livro apresenta a seleção de Minicursos da 44ª edição do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), realizado na Praia do Forte (BA), de 25 a 29 de maio de 2026. O 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. A reconhecida qualidade dos textos produzidos pelos autores dos minicursos tem elevado estes textos à categoria de documentos de referência para trabalhos acadêmicos e formação complementar de estudantes, pesquisadores e profissionais.

Em 2026, 15 propostas de minicursos foram submetidas, um número expressivo que demonstra a importância deste evento no panorama nacional de pesquisa. Destas, 5 foram selecionadas para publicação e apresentação, representando assim uma taxa de aceitação de aproximadamente 33%. O comitê de avaliação dos minicursos foi composto por 19 renomados pesquisadores para a elaboração dos pareceres. Cada proposta recebeu ao menos 3 pareceres, gerando ao todo 45 revisões.

Este livro reúne 5 capítulos produzidos pelos autores das propostas aceitas. O Capítulo 1 discute a fundo a segurança no Aprendizado Federado, explorando e simulando na prática vulnerabilidades e ataques de envenenamento e estratégias modernas de defesa. O Capítulo 2 explora a arquitetura e os desafios do aprendizado colaborativo em ambientes veiculares, apresentando estratégias de rede e novos paradigmas de inteligência artificial para lidar com a latência, a mobilidade e a escassez de dados rotulados. O Capítulo 3 concentra-se no desenvolvimento de agentes inteligentes para a automação e o gerenciamento de redes, integrando conceitos tradicionais de administração de sistemas Linux a tecnologias avançadas de modelos de linguagem. O Capítulo 4 aborda os desafios de engenharia por trás das redes seguras contra a ameaça da computação quântica, focando na gestão de chaves e na integração prática da Distribuição Quântica de Chaves (QKD) com as infraestruturas clássicas existentes. Por fim, o Capítulo 5 introduz os conceitos fundamentais e os desafios arquiteturais dos gêmeos digitais, demonstrando de forma prática como construir, conectar e sincronizar essas entidades virtuais e físicas utilizando o *middleware* MidDiTS.

Como Coordenadores de Minicursos, expressamos nosso sincero agradecimento aos membros do comitê de avaliação pela participação voluntária e pelo excelente trabalho realizado no processo de avaliação e seleção dos minicursos. Agradecemos também aos coordenadores gerais do SBRC 2026, Leobino Nascimento Sampaio (UFBA) e Allan Edgard Silva Freitas (IFBA), pela disponibilidade, pelo suporte constante ao longo de todo o processo e pela confiança depositada em nós para a coordenação. Finalmente, agradecemos aos autores por prestigiarem o SBRC com a submissão de suas propostas e por apresentarem seus minicursos.

Anelise Munaretto e Igor Monteiro Moraes
Coordenadores de Minicursos do SBRC 2026

Comitê de Avaliação de Minicursos

Alberto Egon Schaeffer-Filho, Universidade Federal do Rio Grande do Sul (UFRGS)
Alex Borges Vieira, Universidade Federal de Juiz de Fora (UFJF)
Alfredo Goldman, Universidade de São Paulo (USP)
Anelise Munaretto, Universidade Tecnológica Federal do Paraná (UTFPR)
Christian Esteve Rothenberg, Universidade Estadual de Campinas (Unicamp)
Debora C. Muchaluat-Saade, Universidade Federal Fluminense (UFF)
Denis Rosário, Universidade Federal do Pará (UFPA)
Dianne Scherly Varela de Medeiros, Universidade Federal Fluminense (UFF)
Eduardo Coelho Cerqueira, Universidade Federal do Pará (UFPA)
Glauber Dias Gonçalves, Universidade Federal do Piauí (UFPI)
Helder Oliveira, Universidade de São Paulo (USP)
Igor Monteiro Moraes, Universidade Federal Fluminense (UFF)
Jussara Almeida, Universidade Federal de Minas Gerais (UFMG)
Leobino Nascimento Sampaio, Universidade Federal da Bahia (UFBA)
Mauro Fonseca, Universidade Tecnológica Federal do Paraná (UTFPR)
Rafael Lopes Gomes, Universidade Estadual do Ceará (UECE)
Rodrigo de Souza Couto, Universidade Federal do Rio de Janeiro (UFRJ)
Ronaldo Alves Ferreira, Universidade Federal do Mato Grosso do Sul (UFMS)
Weverton Luis da Costa Cordeiro, Universidade Federal do Rio Grande do Sul (UFRGS)

Minicursos

Capítulo 1

Ataques em Aprendizado Federado: Impactos Práticos e Estratégias de Mitigação

Helio N. Cunha Neto (UERJ), Carlos Henrique Nunes (UERJ),
Ricardo Lundgren (UFF), Raphael Jorge B. Ortolan (UERJ),
Luiz H. S. Ladeira (UERJ), Ian Vilar Bastos (UERJ),
Evandro L. C. Macedo (UERJ), Rafaela C. Brum (UERJ),
Alexandre Sztajnberg (UERJ), Diogo M. F. Mattos (UFF)

Abstract

This chapter presents the theoretical and practical foundations of security in Federated Learning (FL), focusing on major attacks that impact the performance and privacy of decentralized models. It examines vulnerabilities such as model poisoning, gradient inference, sample reconstruction with Generative Adversarial Networks (GAN), model inversion, and backdoor attacks, along with mitigation strategies like robust aggregation (median, trimmed mean, Krum), differential privacy, and homomorphic encryption. Participants will use the Flower framework to implement and evaluate attacks and defenses in simulated FL environments, applying reproducible code to assess the effectiveness of countermeasures.

Resumo

Este capítulo apresenta os fundamentos teóricos e práticos de segurança em Aprendizado Federado (Federated Learning - FL), com foco nos principais ataques que afetam o desempenho e a privacidade de modelos descentralizados. São examinadas vulnerabilidades como envenenamento de modelo, inferência de gradientes, reconstrução de amostras com Redes Adversariais Generativas (GAN), inversão de modelo e ataques backdoor, além de estratégias de mitigação como agregação robusta (median, trimmed mean, Krum), privacidade diferencial e criptografia homomórfica. Os participantes utilizarão o arcabouço Flower para implementar e avaliar ataques e defesas em ambientes simulados de FL, aplicando códigos reproduzíveis para avaliar a efetividade das contramedidas.

Este capítulo foi realizado com recursos do CNPq, CAPES - Código de Financiamento 001, RNP e FAPERJ. Ferramentas de Inteligência Artificial Generativa, incluindo ChatGPT, Grammarly e Perplexity, foram empregadas na revisão textual deste trabalho.

1.1. Introdução

Aprendizado Federado (*Federated Learning* – FL) é um paradigma de aprendizado de máquina decentralizado no qual múltiplos clientes treinam colaborativamente um modelo sem a necessidade de centralizar os dados brutos em um único repositório [Brendan McMahan et al., 2017]. Nesse paradigma, os dados permanecem localmente nos dispositivos ou nas organizações de origem. Cada cliente treina um modelo local com seus próprios dados e compartilha com um terceiro confiável, o servidor agregador, apenas atualizações de parâmetros, gradientes ou estatísticas agregadas [Lim et al., 2020]. Essa característica torna o Aprendizado Federado especialmente relevante em cenários nos quais privacidade, confidencialidade, soberania de dados e restrições regulatórias limitam a coleta centralizada de informações [Cunha Neto et al., 2023]. Aplicações embarcadas, sistemas móveis e serviços personalizados em larga escala têm demonstrado interesse pelo Aprendizado Federado como alternativa para treinamento descentralizado sem compartilhamento de dados dos usuários.

Iniciativas contínuas de pesquisa em Aprendizado Federado estão presentes na academia, literatura científica e têm sido conduzidas por grandes empresas de tecnologia. A Apple tem reportado arquiteturas de *private federated learning* para treinamento em dispositivos de borda e sistemas federados voltados à personalização e ao aprendizado em escala [Ji et al., 2025]. O Google, por sua vez, tem destacado implantações de Aprendizado Federado com garantias formais de privacidade diferencial em produção, inclusive em aplicações do Gboard [Hard et al., 2018], além de continuar expandindo técnicas de análise e adaptação federada em ambientes reais[1]. Em paralelo, a NVIDIA mantém o ecossistema NVIDIA FLARE como plataforma aberta para colaboração distribuída e treinamento federado em múltiplos domínios, incluindo aplicações recentes em dispositivos móveis e modelos de maior porte[2]. Em conjunto, essas iniciativas indicam que o Aprendizado Federado continua sendo uma linha tecnológica estratégica, impulsionada por demandas concretas de treinamento sobre dados descentralizados e por desafios ainda em aberto relacionados a eficiência, robustez e privacidade. Arcabouços como PySyft[3], Flower[4] e TensorFlow Federated (TFF)[5] têm desempenhado papel fundamental na viabilização do Aprendizado Federado.

Apesar desses benefícios, o Aprendizado Federado não é intrinsecamente seguro [Lyu et al., 2020]. A sua característica descentralizada e a consequente ausência de controle centralizado absoluto ampliam sua superfície de ataque, uma vez que os dispositivos clientes, agora incorporados ao processo de treinamento, podem introduzir vulnerabilidades adicionais [Cunha Neto et al., 2023]. Nesse contexto, os ataques podem ter finalidades distintas: busca por extrair informações sensíveis, como dados de treinamento, gradientes ou propriedades estatísticas dos conjuntos locais, ou a tentativa de comprometimento direto do processo de aprendizado, degradando a convergência, reduzindo a acurácia do modelo global ou inserindo comportamentos maliciosos específicos [Liu et al.,

[1]Disponível em https://research.google/blog/federated-learning-with-formal-differential-privacy-guarantees/. Acessado em 21/04/2026

[2]Disponível em https://developer.nvidia.com/flare. Acessado em 24/04/2026

[3]Disponível em https://docs.openmined.org/en/latest/. Acessado em 21/04/2026

[4]Disponível em https://flower.ai/. Acessado em 21/04/2026

[5]Disponível em https://www.tensorflow.org/federated?hl=pt-br. Acessado em 21/04/2026

2022]. Assim, surgem diferentes classes de ataque ao FL, incluindo ataques de envenenamento de modelo (*model poisoning*), envenenamento de dados, ataques de inversão de modelo, reconstrução baseada em redes adversárias generativas (*Generative Adversarial Networks* – GAN), ataques bizantinos e ataques *backdoor*. Em conjunto, esses ataques podem comprometer tanto a confidencialidade dos dados dos clientes quanto a integridade e o desempenho do modelo global [Liu et al., 2022]. Assim, em contextos sensíveis como saúde, finanças, cidades inteligentes e Internet das Coisas (*Internet of Things* – IoT), a proteção dos dados torna-se ainda mais importante, pois a exposição ou vazamento de dados e informações pode comprometer a segurança dos usuários, violar legislações de privacidade cada vez mais rigorosas e até mesmo atentar contra vidas [Bochie et al., 2021].

Este capítulo apresenta uma abordagem teórico-prática, propondo a implementação e análise experimental de diferentes estratégias de ataques. São conduzidos experimentos em um ambiente de Aprendizado Federado baseado no arcabouço Flower[6]. A fim de motivar os participantes e investigar a viabilidade de inferência de dados a partir de informações compartilhadas durante o treinamento, os participantes realizarão a implementação dos seguintes ataques:

- ***Deep Leakage from Gradients* (DLG):** neste ataque há a reconstrução de dados privados de um cliente ao ajustar entradas fictícias até que seus gradientes coincidam com os gradientes compartilhados no Aprendizado Federado.
- **Reconstrução com GAN:** nesse ataque, o atacante utiliza o modelo global como discriminador para treinar um gerador capaz de sintetizar imagens com características semelhantes às amostras presentes no conjunto de dados dos demais participantes, em especial da classe alvo que se deseja inferir.
- **Bizantinos:** neste ataque, clientes maliciosos enviam atualizações aleatórias ou manipuladas ao servidor de agregação, com o objetivo de degradar o desempenho do modelo global. Nessa etapa, serão implementados três ataques bizantinos, ruído gaussiano, *A Little is Enough* (ALIE) e inversão de sinal.

O restante do capítulo está organizado da seguinte forma. A Seção 1.2 apresenta a fundamentação teórica sobre o funcionamento do Aprendizado Federado. A Seção 1.3 discute os principais ataques, enquanto a Seção 1.4 expõe as estratégias de mitigação aos ataques. A atividade prática é detalhada na Seção 1.5 e a Seção 1.6 analisa as tendências de pesquisa e desafios futuros. A Seção 1.7 conclui o capítulo.

1.2. Fundamentação Teórica de Aprendizado Federado

O Aprendizado Federado é um paradigma de treinamento colaborativo em que um modelo global é aprendido a partir de dados mantidos em dispositivos ou instituições distribuídas, sem a necessidade de centralizar os dados brutos. Nesse processo, cada participante treina localmente uma cópia do modelo com seus próprios dados e transmite

[6]Disponível em https://flower.ai/. Acessado em 22/04/2026.

apenas as atualizações de parâmetros ao servidor agregador, preservando a confidencialidade dos conjuntos de dados. A cada rodada, o servidor coleta essas contribuições, combina as atualizações recebidas e produz uma nova versão do modelo global, que é redistribuída aos participantes para continuidade do treinamento, mostrado na Figura 1.1.

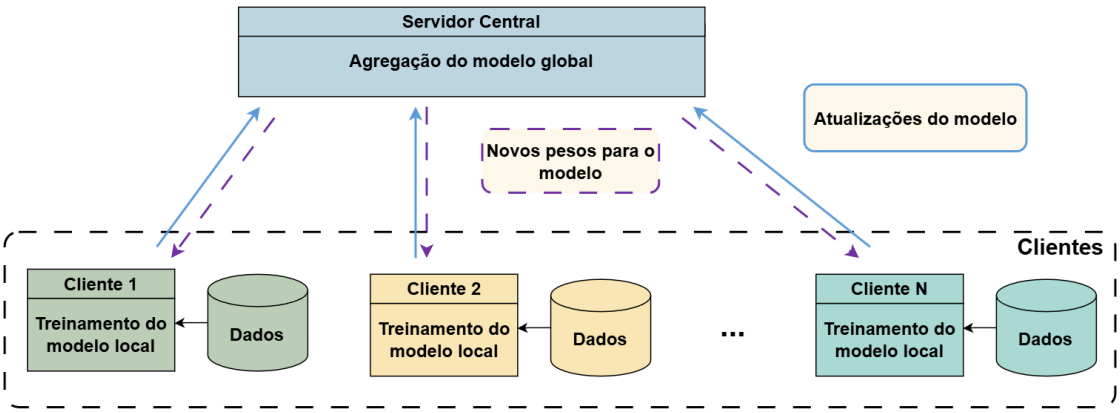


Figura 1.1. Fluxo geral do Aprendizado Federado em arquitetura centralizada. (i) O servidor envia os parâmetros atuais do modelo global aos clientes. (ii) Cada cliente realiza o treinamento local sobre seus próprios dados e retorna apenas atualizações do modelo. (iii) O servidor agrega essas contribuições para gerar um novo modelo global, sem a centralização de dados brutos dos participantes.

A geração do modelo global no Aprendizado Federado ocorre por meio de rodadas de comunicação em que se alternam treinamento local e agregação de parâmetros. Em cada rodada, o processo pode ser organizado em três etapas: (i) o servidor central seleciona os clientes participantes, que podem corresponder a uma fração ou à totalidade dos clientes, e envia a eles os parâmetros atuais do modelo; (ii) cada cliente escolhido atualiza seu modelo local com esses parâmetros, executa o treinamento por algumas épocas com seus dados privados e devolve ao servidor os parâmetros atualizados; (iii) ao receber as atualizações de todos os clientes participantes, o servidor agrega os parâmetros utilizando algoritmos específicos, como o FedAvg [McMahan et al., 2016], obtendo o modelo global daquela rodada de comunicação.

A dinâmica, mostrada na Figura 1.1, caracteriza a arquitetura de Aprendizado Federado centralizado e síncrono, na qual um servidor central coordena a agregação das atualizações dos clientes aguardando o recebimento de todos os parâmetros esperados [Zhang et al., 2025, Pang et al., 2025]. Nesse cenário, o servidor atua como ponto de coordenação e de decisão, o que simplifica o controle do processo de treinamento, mas também cria um ponto único de falha e potencial gargalo de comunicação. A partir dessa formulação centralizada e síncrona, variações da dinâmica original de Aprendizado Federado levam a arquiteturas alternativas, como o Aprendizado Federado descentralizado [Zhang et al., 2025] e o Aprendizado Federado assíncrono ou semi-assíncrono [Pang et al., 2025], que buscam mitigar limitações de robustez e escalabilidade.

O Aprendizado Federado descentralizado [Zhang et al., 2025] remove o papel do servidor central de agregação e distribui a responsabilidade de combinação dos modelos entre os próprios clientes (Figura 1.2). Nessa arquitetura, todos os clientes participam de todas as rodadas de comunicação e, ao final de seus treinamentos locais, enviam seus pa-

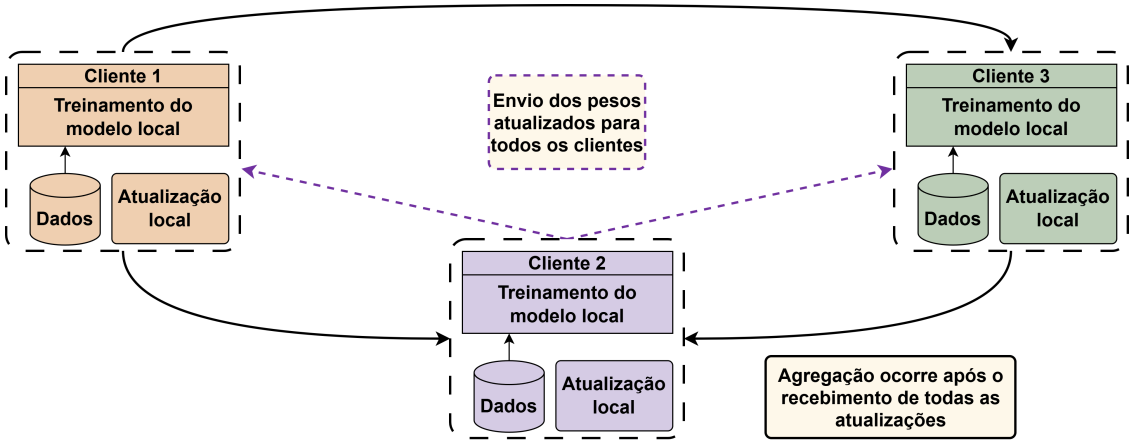


Figura 1.2. Arquitetura do Aprendizado Federado Descentralizado. Não há servidor central: cada cliente treina localmente, compartilha atualizações com seus vizinhos e realiza agregação distribuída. Essa abordagem elimina o ponto único de falha, ao custo de maior complexidade de comunicação e coordenação.

râmetros atualizados diretamente aos demais clientes, que então realizam uma agregação local, em geral utilizando algoritmos simples, como o gradiente descendente estocástico [Zhang et al., 2025]. Essa abordagem elimina o ponto único de falha associado ao servidor central, mas aumenta significativamente o uso da rede, pois o número de mensagens trocadas cresce rapidamente com o número de clientes, impondo novos desafios de escalabilidade e coordenação.

Duas outras variações relevantes, o Aprendizado Federado assíncrono e o semi-assíncrono, podem ser construídas tanto sobre a arquitetura centralizada quanto sobre a descentralizada [Pang et al., 2025]. Nesses casos, a principal diferença em relação ao paradigma síncrono aparece na etapa de agregação, representada pela etapa (iii) da descrição, modificando o momento e a forma como as atualizações dos clientes são incorporadas ao modelo global. Assim, a definição de como e quando agregar as contribuições de cada cliente permite controlar o compromisso entre o quão novas são as atualizações, a estabilidade do treinamento e o custo de comunicação.

No Aprendizado Federado síncrono, a agregação dos parâmetros é realizada apenas depois que o servidor central, ou o conjunto de clientes no caso descentralizado, recebe as atualizações de todos os clientes participantes da rodada. Já no Aprendizado Federado assíncrono, a agregação é executada imediatamente a cada chegada de novos parâmetros, o que tende a aumentar o número de mensagens enviadas pelo servidor para os clientes com versões atualizadas do modelo e pode provocar congestionamento na rede [Pang et al., 2025]. Esse comportamento tende a reduzir o impacto das atualizações de clientes mais lentos, que frequentemente treinam sobre versões defasadas do modelo global, afetando a convergência e a qualidade final do modelo [Pang et al., 2025]. Diante desses efeitos, surge o Aprendizado Federado semi-assíncrono, no qual o servidor (ou cada cliente) realiza a agregação apenas a cada x atualizações recebidas, com x menor que o número total de clientes, ou após um intervalo de m segundos, buscando equilibrar a frequência de atualização e a robustez frente a clientes lentos.

O Aprendizado Federado semi-assíncrono procura reduzir o excesso de atualizações de parâmetros observado no regime totalmente assíncrono, sem introduzir os atrasos

característicos do regime síncrono causados por poucos clientes lentos. Ao limitar a frequência de agregação por número de atualizações ou por janela de tempo, essa abordagem diminui o volume de mensagens e suaviza a variabilidade entre as versões de modelo vistas pelos clientes, preservando parte da eficiência comunicacional do assíncrono.

Independentemente da arquitetura adotada, o Aprendizado Federado pode ser classificado em dois tipos principais, a partir da distribuição das amostras e dos atributos entre os clientes [Yang et al., 2019]. Quando os clientes compartilham o mesmo conjunto de atributos para suas amostras, o cenário é caracterizado como Aprendizado Federado Horizontal; por outro lado, quando os cada cliente observa um subconjunto distinto de atributos das mesmas ou de diferentes amostras, o cenário é definido como Aprendizado Federado Vertical [Yang et al., 2019]. Essa distinção conceitual orienta tanto o desenho dos protocolos de treinamento quanto os mecanismos de privacidade e de comunicação adequados a cada contexto.

No contexto específico do Aprendizado Federado Horizontal, o sistema pode ser ainda categorizado como Aprendizado Federado entre Dispositivos (*Cross-Device Federated Learning*) ou entre Silos de Dados (*Cross-Silo Federated Learning*), de acordo com o tipo de cliente envolvido [Kairouz e McMahan, 2021]. No primeiro caso, os clientes são tipicamente dispositivos de menor capacidade, como celulares e sensores IoT, sujeitos a conectividade de rede instável e recursos computacionais limitados, o que exige estratégias de participação parcial e tolerância a falhas. No segundo caso, os clientes correspondem, em geral, a organizações ou servidores com maior poder computacional e conectividade mais estável, permitindo rodadas de comunicação mais robustas e com maior volume de dados.

1.3. Ataques em Aprendizado Federado

O Aprendizado Federado (FL) está sujeito a diversas ações externas que podem comprometer tanto o desempenho do modelo global quanto a privacidade dos dados dos clientes, usualmente caracterizadas como ataques [Cunha Neto et al., 2023]. Esses ataques são, em geral, agrupados em duas categorias principais: (i) aqueles que visam degradar ou manipular o desempenho do modelo e (ii) aqueles cujo objetivo é violar ou inferir informações sobre os dados privados dos clientes.

A Figura 1.3 mostra em quais etapas do processo federado podem ocorrer os principais vetores de ataque, bem como os mecanismos de defesa associados a essas etapas. Ataques de interceptação podem ocorrer no canal de comunicação, permitindo que um adversário observe atualizações trocadas entre clientes e servidor e, a partir delas, realize inferências sobre informações dos dados locais. No nível dos dados, o envenenamento de dados consiste na manipulação do conjunto utilizado no treinamento local, enquanto o envenenamento do modelo corresponde à alteração direta das atualizações enviadas ao servidor, com o objetivo de influenciar o modelo global. Como contramedidas, a privacidade diferencial e a criptografia homomórfica atuam sobre a informação compartilhada durante a comunicação, reduzindo o risco de inferência a partir de atualizações interceptadas ou observadas. Já a agregação robusta atua no servidor, buscando limitar o impacto de atualizações maliciosas no processo de treinamento.

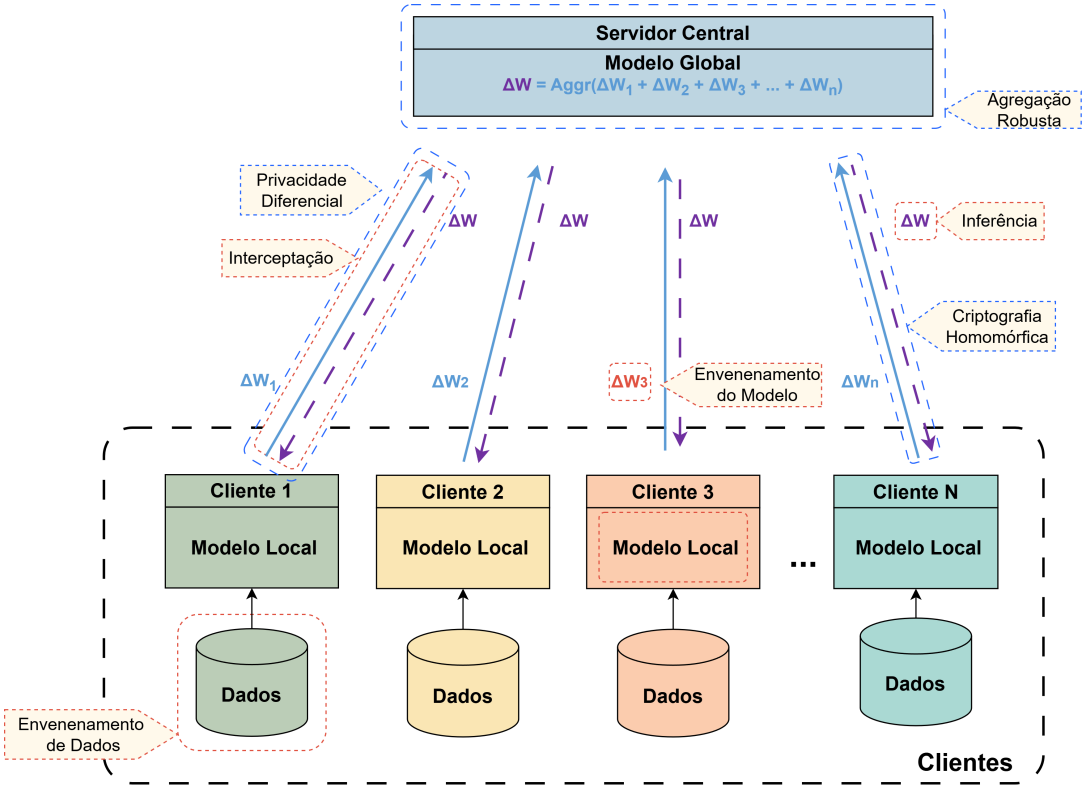


Figura 1.3. Superfícies de ataque e contramedidas no Aprendizado Federado. A figura mostra o fluxo entre servidor, clientes e dados locais, destacando ataques no canal de comunicação, no treinamento local e nas atualizações enviadas ao servidor. Também são indicadas defesas associadas a essas etapas, como privacidade diferencial, criptografia homomórfica e agregação robusta.

1.3.1. Ataques ao Desempenho do Modelo

O desempenho do modelo global em uma abordagem de Aprendizado Federado pode ser comprometido por clientes maliciosos, pois o servidor de agregação observa apenas as atualizações enviadas após o processamento local, sem visibilidade direta sobre o funcionamento interno que as gerou [Lim et al., 2020]. Nesse contexto, a literatura mostra que o modelo pode ser corrompido tanto por meio da adulteração do conjunto de dados utilizado no treinamento local [Goldblum et al., 2022] quanto pela manipulação explícita de gradientes ou parâmetros dos modelos dos clientes [Liu et al., 2022]. O desempenho global também pode ser afetado por ataques com finalidades distintas, como ataques bizantinos, voltados à degradação do treinamento, e ataques *backdoor*, voltados à inserção de comportamentos maliciosos específicos no modelo resultante.

Como essas ameaças diferem também na forma de execução, nesta seção as discussões são organizadas em três dimensões complementares. Em primeiro lugar, consideram-se as características operacionais do ataque, incluindo comportamento adaptativo, furtividade, operação descentralizada e execução assíncrona. Em seguida, analisa-se a finalidade do ataque, distinguindo-se ataques bizantinos de ataques *backdoor*. Por fim, avalia-se a superfície de manipulação, separando envenenamento do conjunto de dados e envenenamento do modelo; essa taxonomia, Figura 1.4, permite relacionar o objetivo do atacante, o ponto de interferência e as propriedades operacionais que determinam o impacto e a dificuldade de detecção de cada ataque.

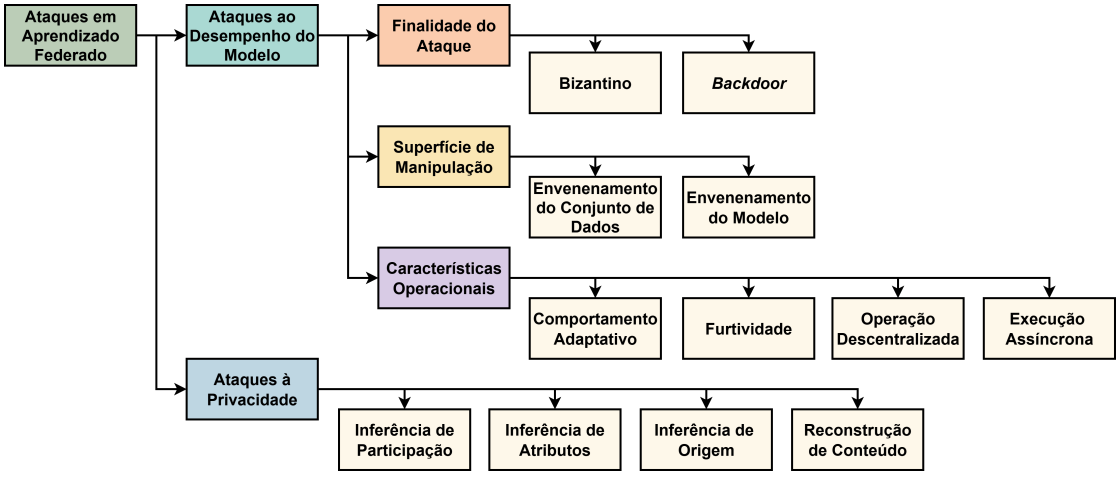


Figura 1.4. Taxonomia dos ataques em Aprendizado Federado. (i) Os ataques ao desempenho do modelo incluem ataques Bizantinos e backdoor, podendo atuar por envenenamento de dados ou do modelo e apresentando diferentes características operacionais, como adaptatividade, furtividade e execução distribuída. (ii) Os ataques à privacidade abrangem técnicas de inferência (participação, atributos e origem) e reconstrução de conteúdo.

Na dimensão de **Características Operacionais**, os ataques são analisados a partir de como são efetivamente executados no ambiente federado, considerando aspectos dinâmicos e estruturais de sua operação. Em particular, ataques com a mesma finalidade e a mesma superfície de manipulação podem diferir quanto ao nível de adaptação aos mecanismos de defesa, ao grau de furtividade buscado, à topologia do sistema em que são formulados e ao regime de sincronização adotado.

O *comportamento adaptativo* caracteriza ataques cuja construção depende explicitamente das propriedades do sistema federado alvo, em vez de se basear em uma perturbação fixa e independente do contexto. Nesses cenários, o adversário ajusta sua estratégia em função de elementos como a regra de agregação, os critérios estatísticos monitorados pela defesa, a dinâmica das rodadas de treinamento e, quando disponível, informações benignas de referência [Shejwalkar e Houmansadr, 2021]. A efetividade do ataque deixa de depender apenas da intensidade da perturbação e passa a depender também da capacidade de explorar as próprias premissas de robustez do sistema federado [Li et al., 2023].

A adaptatividade dos ataques pode se manifestar em diferentes níveis, a começar pela própria formulação da atualização maliciosa frente ao mecanismo de agregação robusta empregado pelo servidor. Em uma primeira forma de adaptação, o atacante constrói a perturbação já levando em conta quais padrões tendem a ser rejeitados pela regra de agregação e quais regiões do espaço de parâmetros permanecem aceitas, moldando o ataque para permanecer dentro dessas regiões [Li et al., 2023]. Em outra forma, a adaptação ocorre em relação às estatísticas observadas pelo sistema de detecção, de modo que a atualização maliciosa é ajustada para se manter compatível com métricas como médias, variâncias ou correlações monitoradas como indicadores de anomalia [Yang et al., 2025]. Essas duas frentes de adaptação abrem espaço para ataques que exploram simultaneamente os limiares de robustez e as métricas de monitoramento.

Em vez de fixar previamente uma única regra de ataque, o adversário pode ainda aprender, rodada após rodada, quais ações produzem maior degradação acumulada do

modelo global, fazendo com que a adaptatividade surja ao longo do próprio processo de treinamento. Nessa formulação, a estratégia maliciosa é atualizada com base na resposta observada do sistema, como em abordagens baseadas em aprendizado por reforço, nas quais o atacante aprende uma política para selecionar dinamicamente a perturbação mais eficaz em cada estado do treinamento federado [Li et al., 2022a]. Em cenários semi-assíncronos, essa adaptação pode incluir também a dimensão temporal, combinando a construção da atualização maliciosa com a escolha do momento e da frequência de participação dos clientes comprometidos [Pang et al., 2025].

A *furtividade* caracteriza ataques projetados para produzir efeito adversarial sem gerar sinais facilmente distinguíveis do comportamento benigno. Em Aprendizado Federado, essa propriedade é particularmente relevante porque o servidor, em geral, não observa os dados locais e frequentemente decide apenas com base nas contribuições agregadas ou em estatísticas resumidas das atualizações recebidas [Yang et al., 2025]. Diferentemente do comportamento adaptativo, que enfatiza o ajuste da estratégia adversarial ao agregador, à defesa ou à dinâmica do treinamento, a furtividade enfatiza a redução da detectabilidade da contribuição maliciosa, mesmo depois que o ataque já está em execução [Lyu et al., 2022]; assim, um ataque pode ser adaptativo sem ser furtivo, quando gera perturbações claramente anômalas, ou pode ser furtivo sem grande adaptatividade, quando apenas preserva uma aparência legítima sem um ajuste sofisticado ao contexto.

Em cenários recentes, a furtividade também passa a incorporar uma dimensão temporal, principalmente em ambientes semi-assíncronos. Nesses casos, a variabilidade introduzida pela obsolescência natural das atualizações pode servir para encobrir o ataque, tornando mais difícil distinguir contribuições benignas defasadas de atualizações adversariais cuidadosamente construídas [Pang et al., 2025]. Essa interação entre atraso, heterogeneidade temporal e comportamento malicioso amplia o espaço de ataque e reforça a importância de mecanismos de detecção que levem em conta a evolução das contribuições ao longo das rodadas.

A furtividade, portanto, não se resume à redução da magnitude da perturbação, mas à capacidade de formular contribuições adversariais compatíveis com os mecanismos de observação e decisão disponíveis no sistema [Lyu et al., 2022]. Em consequência, ataques furtivos tendem a ser especialmente desafiadores, pois exploram a lacuna entre causar dano e ser detectado, preservando a aparência de legitimidade da contribuição maliciosa ao longo de várias rodadas de treinamento [Yang et al., 2025].

A *operação descentralizada* caracteriza ataques formulados para ambientes de Aprendizado Federado em que não há um servidor central confiável responsável pela agregação das contribuições locais. Nesses cenários, cada cliente passa a trocar informações diretamente com um conjunto de vizinhos e atualiza seu modelo a partir de interações distribuídas ao longo da topologia de comunicação [Zhang et al., 2025]. Como consequência, a superfície de ataque deixa de se concentrar em um único ponto e passa a ser distribuída pela rede, permitindo que contribuições maliciosas se propaguem localmente e afetem sucessivas etapas de atualização [Fang et al., 2024].

Essa mudança altera a lógica do ataque, pois o adversário deixa de formular uma única contribuição maliciosa voltada a influenciar diretamente um agregado global e passa a explorar a própria estrutura do grafo de comunicação. Nesse contexto, o atacante disse-

mina modelos locais manipulados aos vizinhos, induzindo desvios progressivos na dinâmica distribuída de otimização, uma vez que a atualização de cada cliente passa a depender tanto do treinamento em seus dados quanto dos modelos recebidos de seu conjunto de vizinhos. Se w_i^t denota o modelo do cliente i na rodada t e $\mathcal{N}_i^t$ representa seu conjunto de vizinhos, a atualização local passa, portanto, a refletir a interação entre o modelo treinado localmente e os modelos recebidos de $\mathcal{N}_i^t$, de modo que a inserção de um modelo malicioso em uma vizinhança estratégica pode contaminar múltiplas trajetórias locais de aprendizado mesmo na ausência de um agregador central [Fang et al., 2024].

Essa característica torna a topologia da rede um componente central do ataque, pois a efetividade da contribuição adversarial passa a depender da conectividade do cliente malicioso, da variação temporal do grafo de comunicação e da ocorrência de desconexões parciais entre vizinhos. Em particular, quando o grafo é dinâmico, a propagação do efeito adversarial deixa de ser uniforme e passa a depender do caminho percorrido pela atualização manipulada, de forma que o ataque atua tanto sobre o conteúdo do modelo adulterado quanto sobre a maneira como clientes maliciosos exploram suas interações com vizinhos para difundir essa informação pela rede [Fang et al., 2024].

Outro aspecto importante é que, em ambientes descentralizados, o afastamento de uma atualização local em relação ao comportamento médio nem sempre decorre de ação maliciosa, especialmente quando o sistema está sujeito a mudança de distribuição (*distributional shift*). Nessa situação, uma contribuição pode parecer anômala por refletir uma alteração legítima no padrão estatístico dos dados do cliente, e não necessariamente um ataque, tornando mais difícil distinguir atualizações benignas não estacionárias de modelos locais maliciosos [Zhang et al., 2025].

Na dimensão de **Finalidade do Ataque**, os ataques são organizados de acordo com o objetivo principal perseguido pelo adversário. Diferentes ataques podem produzir efeitos distintos sobre o processo de treinamento colaborativo, seja influenciando a dinâmica de convergência, seja alterando o comportamento do modelo treinado. Alguns ataques têm como objetivo comprometer o treinamento colaborativo como um todo, degradando a convergência ou a qualidade final do modelo global, enquanto outros buscam inserir comportamentos maliciosos específicos que são acionados apenas sob determinadas condições de entrada [Cunha Neto et al., 2023]. Entre as diferentes finalidades de ataques ao FL, podem-se citar os atauqes Bizantinos e os ataques de *backdoor*.

Ataques Bizantinos correspondem a uma ameaça substancial ao treinamento colaborativo em FL [Shi et al., 2022]. Nesse tipo de cenário, clientes maliciosos podem se desviar arbitrariamente do protocolo esperado e enviar atualizações inconsistentes, manipuladas ou totalmente adversariais ao processo de agregação, tornando a identificação dos agentes maliciosos uma tarefa complexa. Diferentemente de uma simples perturbação aleatória, esse tipo de ataque é concebido para prejudicar o processo de agregação e induzir atualizações discrepantes em relação ao objetivo global de otimização, degradando a acurácia do modelo global e, em casos extremos, comprometendo a própria convergência do processo iterativo [Li et al., 2023, Sattler et al., 2020].

Ataques Bizantinos são uma ameaça central em Aprendizado Federado porque o atacante não precisa necessariamente produzir uma atualização evidentemente anômala para causar dano. Assim, o problema deixa de ser apenas a remoção de valores extremos

e passa a incluir a identificação de atualizações que, embora compatíveis com a distribuição observada, foram construídas para enviesar gradualmente a trajetória de otimização. Trabalhos recentes [Li et al., 2023, Fang et al., 2020, Shi et al., 2022] mostram que diversos ataques são projetados para contornar defesas *Byzantine-robust* e indicam que a robustez efetiva depende não apenas do agregador, como da distribuição estatística dos dados entre os clientes, do número de atacantes e da capacidade adaptativa do adversário.

A furtividade pode ser alcançada por diferentes estratégias de construção da atualização maliciosa. Em abordagens mais simples, o atacante pode, por exemplo, inverter o sinal do gradiente ou ampliar artificialmente sua magnitude, produzindo uma atualização que empurra o modelo global na direção oposta àquela esperada no treinamento benigno [Shi et al., 2022]. Em ataques mais eficazes, porém, a atualização maliciosa é construída com base na estatística observada das contribuições benignas. No ataque *A Little Is Enough* (ALIE), por exemplo, clientes maliciosos imitam a dispersão estatística das atualizações benignas ao estimar, para cada componente do vetor de atualização local, a média μ_i e o desvio padrão δ_i e escolher o valor corrompido dentro do intervalo $(\mu_i - z^{\max}\delta_i, ; \mu_i + z^{\max}\delta_i)$, em que $z^{\max}$ controla o afastamento admissível em unidades de desvio-padrão [Li et al., 2023]. Outra técnica consiste em alinhar a atualização maliciosa na direção oposta à média benigna, por exemplo definindo a atualização adversarial como uma versão escalada de $-\bar{g}$, em que $\bar{g}$ é o gradiente benigno médio, o que torna negativo o produto interno entre a direção agregada e o gradiente benigno e impede a descida adequada da função de perda [Li et al., 2023].

Há também ataques formulados explicitamente como problemas de otimização. Nesses ataques, o adversário escolhe uma direção de perturbação, um vetor que especifica como a atualização maliciosa deve se desviar de uma atualização benigna de referência no espaço de parâmetros, e ajusta as contribuições comprometidas para maximizar a diferença entre os gradientes agregados sob ataque e os gradientes agregados em um cenário benigno [Shejwalkar e Houmansadr, 2021, Fang et al., 2020]. Uma forma simplificada de representar esse tipo de ataque é definir a atualização maliciosa como $g_m = g_{\text{ref}} + p$, em que g_{ref} é uma atualização com aparência benigna e p é um vetor de perturbação que introduz o desvio adversarial, sendo o objetivo maximizar $|\tilde{g}(p) - \bar{g}|$, em que $\tilde{g}(p)$ é o gradiente agregado sob ataque e $\bar{g}$ é o gradiente agregado benigno. Para preservar a furtividade, impõem-se restrições que impedem que g_m se afaste excessivamente do conjunto benigno, por exemplo por meio de limites de distância em relação ao centro das atualizações benignas ou de intervalos coordenados definidos a partir da média e do desvio padrão observados em cada parâmetro [Shejwalkar e Houmansadr, 2021, Fang et al., 2020].

Outro aspecto recorrente é que o ataque bizantino não se restringe a cenários com servidor agregador centralizado. Em arquiteturas descentralizadas, a superfície de ataque se amplia, pois uma atualização maliciosa pode se propagar localmente pela rede e influenciar sucessivas etapas de agregação sem a presença de um servidor confiável [Fang et al., 2024]. Nesses cenários, clientes maliciosos podem disseminar modelos locais manipulados para os vizinhos, comprometendo a convergência do sistema e o desempenho final dos modelos aprendidos.

Além dos ataques maliciosos, ambientes descentralizados também estão sujeitos a efeitos de mudança de distribuição (*distributional shift*), que geram ruptura entre a dis-

tribuição dos dados de treinamento e a dos dados observados posteriormente em operação [Zhang et al., 2025]. Nesses casos, a degradação do modelo não decorre de comportamento adversarial, mas de alterações reais no padrão estatístico dos dados, de modo que uma atualização local pode se afastar do comportamento médio tanto por refletir uma mudança legítima na distribuição quanto por ter sido manipulada. Essa sobreposição entre heterogeneidade estatística e comportamento adversarial é crítica em Aprendizado Federado descentralizado, pois dificulta a distinção entre atualizações benignas não-estacionárias e modelos locais maliciosos [Zhang et al., 2025, Fang et al., 2024].

Há ainda formulações mais gerais em que o Ataque Bizantino é modelado de forma probabilística. Nesses modelos, assume-se um adversário onisciente capaz de comprometer apenas uma fração limitada da informação disponível e, ainda assim, escolher a perturbação mais danosa sob uma restrição explícita de esforço [Wang et al., 2025]. Esse adversário é tratado como um agente que conhece o classificador utilizado e o fenômeno verdadeiro subjacente aos dados, mas pode corromper apenas uma fração α das amostras, configurando uma hipótese adversarial forte, mais adequada à análise de pior caso do que à descrição de cenários operacionais típicos [Wang et al., 2025]. Essa perspectiva mostra que a força do ataque pode ser analisada não só pelo número de clientes comprometidos, mas também pelo grau de controle exercido sobre a distribuição das contribuições maliciosas e pelo custo para induzir degradação mensurável no desempenho de classificação.

O *ataque backdoor*, em aprendizado de máquina, é um ataque direcionado em que o adversário faz o modelo manter bom desempenho na tarefa principal, mas induz saídas incorretas sempre que a entrada contém um gatilho específico. Um exemplo clássico é um pequeno padrão fixo inserido em imagens, como um quadrado branco em um canto, que aciona uma predição equivocada apenas quando presente. Assim, o modelo aparenta funcionar normalmente na maioria dos casos, mas incorpora um comportamento malicioso latente ativado sob condições específicas, e, em Aprendizado Federado, esse ataque não precisa degradar visivelmente o desempenho global para ser bem-sucedido, pois sua efetividade depende justamente de preservar a utilidade aparente do modelo na tarefa principal enquanto introduz uma falha controlada em um subconjunto de entradas [Lyu et al., 2022, Wang et al., 2020, Cao et al., 2021].

Uma forma clássica de concretizar o ataque em Aprendizado Federado consiste em aproximar o modelo global do modelo treinado pelo atacante, estratégia frequentemente descrita como substituição do modelo (*model replacement*). Em sistemas baseados em média ponderada, como o FedAvg [McMahan et al., 2016], o adversário explora o fato de que, em cada rodada, o servidor envia o modelo global corrente w^t aos clientes selecionados, de modo que um cliente malicioso i_0 conhece esse estado e não precisa saber exatamente as atualizações dos demais. Assumindo que clientes honestos retornam modelos próximos de w^t, o atacante pode reescalar sua atualização e enviá-la como

$$\frac{n_S}{n_{i_0}}(w_{i_0} - w^t) + w^t, \tag{1}$$

em que n_{i_0} é o tamanho do conjunto de dados local do atacante e $n_S = \sum_{i \in S} n_i$ é o total de dados dos clientes selecionados; como a atualização global em FedAvg é dada por

$$w^{t+1} = w^t + \sum_{i \in S} \frac{n_i}{n_S}(w_i - w^t), \tag{2}$$

se os clientes honestos estiverem próximos de convergência, com $w_i \approx w^t$ para $i \neq i_0$, suas contribuições praticamente se anulam e a atualização reescalada do atacante passa a dominar a agregação, resultando em $w^{t+1} \approx w_{i_0}$ [Wang et al., 2020]. Nessa situação, o comportamento indevido não surge de ruído acumulado, mas de uma atualização planejada, construída para substituir, total ou parcialmente, a trajetória de aprendizado benigno.

Trabalhos mais recentes mostram, porém, que o ataque pode ser implementado de forma mais sutil do que uma simples substituição agressiva do modelo. Ao invés de enviar uma atualização evidentemente discrepante, o atacante pode otimizar localmente seu modelo malicioso impondo restrições para que ele permaneça próximo do modelo global corrente a cada rodada, tornando mais difícil a detecção por agregadores robustos. No caso dos *backdoors* por casos raros (*edge-case backdoors*), essa estratégia é combinada com o uso de amostras pertencentes à cauda da distribuição de entrada, exemplos semanticamente válidos, mas pouco frequentes no conjunto de dados, como por exemplo imagens reais de aviões de uma companhia específica rotuladas incorretamente como "caminhão" em um problema de classificação de imagens [Wang et al., 2020]. Nessa configuração, o gatilho deixa de ser um padrão artificial evidente e passa a ser associado a entradas raras, que tendem a aparecer com baixa probabilidade durante o treinamento e a avaliação convencional, o que reduz a chance de que o comportamento malicioso seja identificado por inspeções empíricas rotineiras.

A literatura também indica que a dificuldade de defesa contra *backdoors* em Aprendizado Federado decorre de limitações estruturais do próprio paradigma [Lyu et al., 2022, Wang et al., 2020]. Como o servidor não observa dados locais e, em cenários com agregação segura, pode ter acesso apenas ao resultado agregado, sem inspecionar individualmente as contribuições dos clientes, muitas defesas tradicionais contra *backdoor* desenvolvidas para aprendizado centralizado tornam-se inaplicáveis. Ademais, em Aprendizado Federado uma defesa precisa ser robusta tanto a manipulações nos dados locais quanto a manipulações diretas nas atualizações de modelo, o que faz com que abordagens focadas principalmente em ataques bizantinos ou em *outliers* não ofereçam necessariamente proteção adequada a ataques direcionados [Lyu et al., 2022]. A robustez a *backdoors* está fortemente associada à robustez a exemplos adversariais. Adicionalmente, a detecção de *backdoors* em modelos treinados pode ser computacionalmente difícil em cenários gerais. Assim, esse cenário reforça que o desafio vai além de projetar filtros melhores para atualizações suspeitas e envolve lidar com uma forma de comprometimento que pode permanecer praticamente invisível, ao mesmo tempo em que preserva a utilidade aparente do modelo na maior parte das entradas [Wang et al., 2020].

Na dimensão de **Superfície de Manipulação**, os ataques são organizados de acordo com a superfície explorada pelo adversário. A manipulação pode incidir sobre os dados que alimentam o treinamento local ou sobre a própria contribuição gerada ao final desse treinamento, como gradientes e parâmetros do modelo. Diferenciar esses tipos de manipulação é importante pois permite separar ataques que alteram indiretamente o comportamento do modelo global, por meio da experiência de treinamento do cliente, daqueles que atuam diretamente sobre a informação enviada ao mecanismo de agregação.

No *envenenamento do conjunto de dados*, o atacante compromete o processo de treinamento a partir da manipulação do conjunto de dados local utilizado no treinamento,

ao invés de atuar diretamente sobre a regra de agregação ou sobre a atualização final enviada ao servidor [Nowroozi et al., 2025]. A manipulação ocorre, portanto, sobre os dados do cliente malicioso, que passam a ser usados em um processo de treinamento aparentemente legítimo, mas baseado em informações corrompidas. Como a atualização local é produzida a partir desses dados adulterados, o efeito adversarial é transferido indiretamente para o modelo global, o que torna essa superfície de ataque especialmente relevante em Aprendizado Federado [Lyu et al., 2022].

Esse tipo de ataque consiste em alterar a distribuição utilizada no treinamento local. Essa alteração pode ocorrer por diferentes mecanismos, como modificação de rótulos, perturbação de características, inserção de amostras sintéticas ou substituição parcial do conjunto original, mas em todos os casos o objetivo final permanece o de influenciar o modelo global. O impacto pode ser tanto na degradação da tarefa principal quanto na indução de comportamentos direcionados em condições de entrada [Wang et al., 2020].

Uma forma particularmente simples de envenenamento é a inversão de rótulos (*label flipping*). O atacante altera o rótulo de parte das amostras locais, mantendo as entradas originais, forçando o modelo a aprender associações inconsistentes entre representação e classe e produzindo gradientes que deixam de apontar para a minimização correta da função de perda [Lyu et al., 2022]. Em sua forma mais direta, o ataque não exige conhecimento profundo sobre a arquitetura ou a regra de agregação, bastando acesso ao conjunto local e à capacidade de modificar as anotações, embora sua efetividade dependa do volume de dados adulterados, da distribuição das classes e da capacidade do treinamento global de absorver o ruído introduzido.

Outra estratégia consiste em manipular características das amostras sem alterar necessariamente seus rótulos. No ataque por envenenamento de características (*feature poisoning*), o atacante modifica atributos considerados influentes para a decisão do modelo, de modo a deslocar a fronteira de decisão sem produzir degradação abrupta e facilmente perceptível no desempenho global. Nowroozi *et al.* mostram o caso em que essa manipulação é guiada por uma floresta aleatória com *permutation feature importance* [Nowroozi et al., 2025], técnica em que os valores de cada característica são embaralhados e o impacto dessa perturbação no desempenho é usado como medida de relevância. Assim, seja I_k a importância estimada da característica k, o atacante seleciona a coluna mais influente $k^\star = \arg\max_k I_k$ e concentra a manipulação nessa dimensão, normalizando seus valores e substituindo a característica, em uma fração P das amostras maliciosas, por valores escolhidos aleatoriamente entre os valores únicos observados [Nowroozi et al., 2025]. Dessa forma, a perturbação incide sobre a variável que mais contribui para a decisão, aumentando a capacidade de deslocar a fronteira de classificação mesmo sem alterar todas as variáveis e mantendo métricas globais aparentemente elevadas [Nowroozi et al., 2025].

Há também ataques em que o envenenamento inclui a geração de novas amostras artificiais. No ataque baseado em Rede Generativa Adversarial (*Generative Adversarial Network* – GAN), o cliente malicioso começa atuando como um participante aparentemente benigno e, ao longo do treinamento federado, utiliza as informações trocadas para treinar localmente uma GAN que aprenda a imitar amostras prototípicas de classes que não estavam originalmente em sua posse [Zhang et al., 2019]. Em seguida, o atacante rotula essas amostras geradas de acordo com seu objetivo malicioso e as insere no trei-

namento local, alterando a composição semântica de seu conjunto de dados e produzindo atualizações que comprometem o modelo global, embora o processo continue se apresentando como um treinamento padrão baseado em dados.

No contexto de ataques direcionados, como *backdoors*, o envenenamento do conjunto de dados costuma ocorrer por inserção parcial de amostras maliciosas no conjunto local. Em vez de substituir completamente a base de treinamento, o atacante preserva exemplos limpos e adiciona amostras contendo o padrão de ativação do ataque, associadas a um rótulo alvo específico, caracterizando o chamado caso *black-box* de inserção de *backdoor* [Wang et al., 2020]. Nessa configuração, o adversário atua apenas sobre os dados, sem uma etapa adicional de otimização explícita da atualização enviada ao servidor: o comportamento malicioso é aprendido pelo modelo local a partir das amostras envenenadas e propagado ao modelo global por meio das atualizações resultantes, de modo que, mesmo quando a finalidade é um *backdoor*, a superfície de manipulação permanece o conjunto de dados [Wang et al., 2020].

Do ponto de vista operacional, o atacante, em envenenamento de dados, depende fortemente da dinâmica do treinamento local para transferir o efeito adversarial ao modelo global. Em alguns cenários, essa dependência torna o ataque menos potente do que abordagens que manipulam diretamente a atualização final enviada ao servidor [Nowroozi et al., 2025, Cunha Neto et al., 2023], pois parte do efeito adversarial pode ser absorvida pelo processo de otimização e pela agregação entre múltiplos clientes. Por outro lado, essa mesma característica tende a torná-lo mais natural e menos suspeito, já que a contribuição local é produzida por uma rotina de treinamento que, do ponto de vista do servidor, se assemelha ao comportamento regular de um cliente benigno [Lyu et al., 2022].

No *envenenamento do modelo*, a manipulação adversarial incide diretamente sobre a atualização que o cliente envia ao servidor agregador, em vez de atuar apenas sobre os dados usados no treinamento local [Lyu et al., 2022]. Em vez de depender do efeito indireto de dados corrompidos, o cliente malicioso constrói explicitamente um modelo adulterado, denotado por w_{i_0}, e o utiliza como vetor adversarial, de modo que o desvio introduzido passa a ser formulado diretamente no objeto matemático agregado pelo servidor [Shi et al., 2022].

Uma forma simples de implementar esse ataque consiste em inverter o sentido da atualização honesta ou ampliar artificialmente sua magnitude. Se a contribuição local legítima do cliente malicioso é $w_{i_0} - w^t$, o atacante pode substituí-la por

$$-\lambda(w_{i_0} - w^t), \qquad \lambda > 0, \tag{3}$$

onde λ é um fator de escala positivo que controla a intensidade do ataque [Shejwalkar e Houmansadr, 2021]. Esse princípio aparece em ataques de inversão de sinal (*sign flipping*) e também em variantes nas quais a magnitude da atualização maliciosa é ampliada para aumentar sua influência sobre a média agregada, induzindo o modelo global a evoluir na direção oposta àquela esperada no treinamento benigno [Shi et al., 2022].

Uma estratégia mais direcionada consiste em manipular explicitamente o alinhamento entre a atualização agregada e a direção benigna de otimização [Li et al., 2023]. Seja $\bar{u}$ a média das atualizações honestas, ou uma estimativa dessa média obtida a partir de contribuições benignas conhecidas pelo atacante, a contribuição maliciosa pode ser

construída na direção oposta, por exemplo $\tilde{u}_{i_0} = -\lambda \bar{u}$, com $\lambda > 0$ [Shejwalkar e Houmansadr, 2021]. Diferentemente da simples inversão da própria atualização local, essa construção visa interferir diretamente na direção média da agregação, buscando tornar negativo ou suficientemente pequeno, o produto interno entre a atualização agregada e a direção benigna e, assim, reduzir o alinhamento do treinamento com a descida da função de perda [Li et al., 2023].

Há também ataques formulados como problemas explícitos de otimização. Nesses casos, o adversário parte de uma atualização de referência u_{ref} e define uma direção de perturbação p, obtendo uma contribuição comprometida da forma

$$\tilde{u}_{i_0} = u_{\text{ref}} + p. \tag{4}$$

O vetor p é então escolhido para maximizar a diferença entre o agregado sob ataque $\tilde{u}_{\text{agg}}(p)$ e o agregado benigno u_{agg}, sob restrições que impeçam que a atualização maliciosa se afaste excessivamente do conjunto benigno [Li et al., 2023]. Em termos genéricos, pode-se escrever

$$\max_{p} \ \left\| \tilde{u}_{\text{agg}}(p) - u_{\text{agg}} \right\| \quad \text{sujeito a} \quad u_{\text{ref}} + p \in \mathscr{C} \tag{5}$$

em que $\mathscr{C}$ representa o conjunto de restrições impostas para reduzir a chance de rejeição pelo agregador, como limites de distância em relação ao centro das atualizações benignas [Shejwalkar e Houmansadr, 2021]. Essa lógica está no núcleo de duas famílias de ataques: os ataques adaptados à regra de agregação (*AGR-tailored*), em que o adversário conhece o mecanismo robusto empregado e o contorna explicitamente, e os agnósticos (*AGR-agnostic*), em que o agregador é desconhecido e a contribuição maliciosa é formulada de modo mais geral, mantendo-se próxima do conjunto benigno [Li et al., 2023].

Outra formulação é o desvio direcionado (*directed deviation*). Nesse caso, o atacante estima a direção s em que o modelo global evoluiria na ausência de ataque e constrói a contribuição comprometida para deslocar cada parâmetro no sentido oposto, como

$$\tilde{u}_{i_0} = u_{i_0} - \gamma s, \tag{6}$$

com $\gamma > 0$ controlando a intensidade do desvio [Fang et al., 2020]. O objetivo não é apenas gerar uma atualização arbitrária, mas interferir na trajetória de otimização de forma coordenada, componente a componente, aumentando a efetividade do ataque contra agregadores que filtram apenas *outliers* grosseiros [Fang et al., 2020].

Modelos ainda mais sofisticados combinam esse objetivo com restrições explícitas de furtividade. Uma estratégia é resolver localmente um problema de otimização multicritério, no qual o atacante minimiza simultaneamente a perda da tarefa maliciosa e um termo de proximidade em relação ao modelo global corrente, por exemplo

$$\min_{w_{i_0}} \ \mathscr{L}_{\text{adv}}(w_{i_0}) + \beta \| w_{i_0} - w^t \|^2, \tag{7}$$

em que $\mathscr{L}_{\text{adv}}$ representa a perda associada ao comportamento adversarial desejado e o termo $\beta \| w_{i_0} - w^t \|^2$ penaliza afastamentos excessivos do estado global [Wang et al., 2020]. Essa formulação permite preservar o efeito do ataque ao mesmo tempo em que

reduz a discrepância entre a contribuição maliciosa e o modelo benigno esperado [Wang et al., 2020].

O envenenamento do modelo também pode ser visto como um problema sequencial de decisão. Em vez de adotar uma heurística fixa, o adversário pode usar aprendizado por reforço (*Reinforcement Learning* – RL) para aprender, ao longo das rodadas, quais ações produzem maior degradação acumulada do modelo global [Li et al., 2022a]. Nesse contexto, o processo de ataque é modelado como uma sequência de interações em que, a cada rodada t, o atacante observa um estado s_t que resume informações relevantes sobre o treinamento, escolhe uma ação a_t conforme uma política parametrizada $a_t \sim \pi_\theta(s_t)$ e recebe um retorno que mede o impacto adversarial obtido [Li et al., 2022a]. O objetivo passa a ser aprender os parâmetros θ de uma política que selecione, em cada rodada, a contribuição maliciosa de maior impacto esperado, inclusive na presença de agregadores robustos [Li et al., 2022a].

O envenenamento do modelo não se restringe ao cenário centralizado síncrono. Em Aprendizado Federado descentralizado, a contribuição maliciosa circula entre vizinhos, permitindo que o ataque explore a topologia de comunicação para propagar atualizações manipuladas [Fang et al., 2024]. Em regimes semi-assíncronos, a defasagem temporal entre atualizações cria oportunidades adicionais, pois clientes distintos treinam sobre versões diferentes do modelo global e suas contribuições chegam ao servidor com diferentes níveis de obsolescência [Pang et al., 2025]. Nesse cenário, o adversário pode explorar não apenas o conteúdo da atualização maliciosa, mas também o momento e a frequência de participação dos clientes comprometidos, como no *framework* PoiSAFL, em que modelos locais envenenados são combinados com controle da frequência de envio para amplificação no processo semi-assíncrono de agregação [Pang et al., 2025].

1.3.2. Ataques à Privacidade

Os ataques à privacidade em Aprendizado Federado também podem ser categorizados pelo tipo de informação privada que o adversário busca extrair a partir do processo de treinamento. Dessa forma, os ataques são agrupados em quatro categorias: (i) ataques de inferência de participação, cujo objetivo é determinar se uma amostra específica integrou o conjunto de treinamento de um cliente [Shokri et al., 2017]; (ii) ataques de inferência de atributos, voltados à extração de propriedades latentes, sensíveis ou não intencionais presentes nos dados privados, mesmo quando não estão diretamente relacionadas à tarefa principal do modelo [Melis et al., 2019]; (iii) ataques de inferência de origem, nos quais o adversário busca associar informações, rótulos ou evidências a uma fonte específica, como um cliente do treinamento federado [Hu et al., 2023]; e (iv) ataques de reconstrução de conteúdo, cujo propósito é recuperar, aproximar ou sintetizar o próprio dado privado a partir de gradientes ou atualizações de modelo [Wu et al., 2024].

Os **ataques de inferência de participação** têm como objetivo decidir se uma amostra específica x pertenceu ou não ao conjunto de treinamento utilizado para ajustar um modelo alvo w. Formalmente, dado um modelo treinado w e uma amostra alvo x, o atacante busca estimar uma variável binária $m(x) \in 0,1$, em que $m(x) = 1$ indica que x foi usada no treinamento e $m(x) = 0$ indica o contrário [Shokri et al., 2017]. Em termos probabilísticos, o ataque consiste em construir um inferidor A capaz de aproximar

$P(m(x) = 1 \mid \psi(x,w))$, em que $\psi(\cdot)$ representa o conjunto de observáveis disponíveis ao adversário, como vetor de probabilidades, perda, ativações internas, gradientes locais ou atualizações federadas [Melis et al., 2019]. A relevância desse tipo de ataque decorre das evidências de participação já constituir uma violação severa de privacidade, mesmo sem revelar explicitamente o conteúdo do registro, em diversos domínios sensíveis.

Na formulação clássica [Shokri et al., 2017], considera-se o cenário de caixa preta (*black-box*) de acesso ao modelo, em que o adversário observa apenas a saída $w(x)$, normalmente expressa como um vetor de probabilidades sobre as classes. O ataque é convertido em um problema supervisionado auxiliar, em que se constrói um modelo atacante A que recebe como entrada a saída do modelo alvo e produz como saída a classe "membro" ou "não membro". Para obter dados rotulados para treinar A, utiliza-se a técnica de modelos-sombra (*shadow models*), na qual o atacante treina modelos auxiliares $w^{(1)}, \ldots, w^{(k)}$ em conjuntos de dados sob seu controle, de modo a reproduzir, de forma aproximada, o comportamento estatístico do modelo alvo. Para cada modelo auxiliar $w^{(i)}$, o atacante sabe exatamente quais amostras foram usadas no treinamento e quais foram mantidas fora, de modo que, ao consultar $w^{(i)}$ com exemplos desses dois grupos, obtém pares rotulados da forma $(w^{(i)}(x), m(x))$, que são então utilizados para treinar o classificador de ataque.

Melis *et al.* mostram que a inferência de participação não se limita a modelos centralizados acessados via *Application Programming Interface* (API), mas também surge naturalmente em aprendizado colaborativo [Melis et al., 2019]. A atualização compartilhada por cada cliente é calculada diretamente a partir de um lote local de dados e, portanto, preserva traços estatísticos desse lote: em uma rodada t, dado o modelo global w^t, cada cliente computa localmente um gradiente $\nabla L(b; w^t)$ sobre um lote b de seus dados e envia essa atualização ao servidor de agregação. Como direção e magnitude desse gradiente dependem explicitamente das amostras presentes em b, a atualização funciona como uma projeção do conteúdo do lote no espaço de parâmetros do modelo, o que permite a um cliente adversário, ao observar gradientes individuais ou diferenças sucessivas entre parâmetros globais, explorar essas variações para inferir se uma determinada amostra esteve presente no treinamento dos demais participantes [Melis et al., 2019].

O vazamento torna-se particularmente claro em camadas de entrada esparsas, como em *embeddings*, nas quais apenas as coordenadas associadas aos *tokens* efetivamente presentes no lote recebem gradientes não nulos, revelando diretamente a ocorrência desses elementos. Mesmo quando o protocolo de Aprendizado Federado expõe apenas o modelo global agregado, o adversário pode explorar a evolução temporal dos parâmetros, pois a diferença $\Delta w^t = w^t - w^{t-1}$ representa o efeito acumulado das atualizações dos participantes selecionados na rodada t. Se o adversário também participa do treinamento, sua própria contribuição local é conhecida e pode ser subtraída de Δw^t, produzindo uma estimativa da atualização agregada gerada pelos demais clientes, que ainda contém traços dos dados usados no treinamento local dos participantes honestos [Melis et al., 2019].

No cenário caixa branca (*white-box*), o adversário não se limita a observar a saída preditiva $w(x)$, mas explora parâmetros internos do modelo avaliados sobre a amostra alvo x. Esses parâmetros incluem ativações intermediárias das camadas ocultas, a perda $\mathscr{L}(w(x), y)$ e, principalmente, o gradiente dessa perda em relação aos parâmetros da rede,

$g(x) = \nabla \mathscr{L}(w(x), y)$. Entre essas quantidades, o vetor gradiente é o mais informativo para inferência de participação, pois expressa quanto e em que direção cada parâmetro ainda precisaria ser ajustado para reduzir a perda associada à amostra x; como o treinamento por gradiente descendente estocástico (*Stochastic Gradient Descent* – SGD) atualiza iterativamente os parâmetros para minimizar a perda sobre exemplos vistos, registros pertencentes ao conjunto de treinamento tendem a induzir padrões de gradiente estatisticamente distintos daqueles produzidos por amostras não-membros, de forma que cada exemplo de treino deixa uma "assinatura" no estado final dos parâmetros, mais evidente no espaço dos gradientes do que apenas na saída final do modelo [Zhu et al., 2025].

Com base nessa observação, Nasr *et al.* propõem ataques de inferência de participação em cenário caixa branca que utilizam gradientes locais por camada como entrada de um modelo atacante [Nasr et al., 2019]. Os autores mostram que mesmo modelos bem generalizados podem permanecer vulneráveis quando o adversário tem acesso aos parâmetros e às quantidades derivadas do processo de otimização. Por exemplo, para verificar se uma amostra x pertence ao conjunto de treinamento, o atacante pode, no cenário *white-box*, avaliar x no modelo treinado, calcular a perda $\mathscr{L}(w(x), y)$ e obter o gradiente local, que indica quanto cada parâmetro ainda precisaria ser ajustado para reduzir a perda associada a x [Xia et al., 2024]. Se x tiver sido usada no treinamento, espera-se que o modelo já esteja parcialmente adaptado a essa amostra, de modo que $g(x)$ apresente um padrão estatístico compatível com amostras membros. Caso contrário, o gradiente tende a refletir maior desalinhamento entre a amostra e o estado final dos parâmetros, sendo então utilizado, em suas componentes por camada, como entrada de um classificador de ataque que estima a probabilidade de x ser membro do conjunto de treinamento.

Os **ataques de inferência de atributos** têm como objetivo extrair propriedades dos conjuntos de dados dos clientes utilizados no treinamento do Aprendizado Federado [Melis et al., 2019]. Tais propriedades podem ser inferidas mesmo quando não constituem o alvo da tarefa principal aprendida pelo modelo global. Diferentemente da inferência de participação, em que o adversário busca decidir se uma amostra específica integrou o conjunto de treinamento, a inferência de atributos procura responder se um lote, subconjunto ou distribuição local de dados apresenta uma propriedade p de interesse [Diana et al., 2025]. Formalmente, considere uma amostra anotada como $z = (x, y, p)$, em que x representa a entrada, y o rótulo da tarefa principal e $p \in \{0, 1\}$ uma propriedade privada latente, potencialmente independente de y; enquanto y define o que o modelo foi treinado para prever, p define o que o atacante deseja descobrir a partir do comportamento do modelo ou das atualizações compartilhadas. Nessa formulação, o adversário busca estimar uma variável do tipo

$$a(b) = \mathbb{I}\{\exists z \in b : p(z) = 1\}, \tag{8}$$

ou, em variantes mais gerais, a fração de exemplos em um lote b que satisfazem p, de modo que o ataque não pretende reconstruir o dado bruto nem decidir sobre a presença de uma amostra específica, mas inferir uma característica semântica emergente do comportamento do modelo frente às atualizações induzidas pelos dados privados.

Enquanto abordagens centradas em classes exploram propriedades que caracterizam uma categoria inteira, a inferência de atributos [Melis et al., 2019] opera sobre propriedades válidas apenas para subconjuntos dos dados e potencialmente independentes da

tarefa alvo. Em vez de tentar inferir como é, em média, a classe y, o adversário procura inferir se os dados usados em uma atualização específica apresentam a propriedade p, o que é relevante porque o vazamento pode ocorrer mesmo quando essa propriedade não contribui para o desempenho preditivo do modelo, mas é capturada como consequência da riqueza estatística dos dados de entrada [Wang et al., 2022]. Embora o modelo seja treinado para a tarefa principal, as atualizações compartilhadas ainda podem carregar sinal suficiente para inferir atributos colaterais dos dados privados, de modo que o canal de vazamento deixa de ser o rótulo principal y e passa a ser a estrutura estatística das representações internas e dos gradientes associados a propriedades que o modelo não deveria, em princípio, revelar [Jayaraman e Evans, 2022].

No ataque passivo, o adversário não altera o protocolo de treinamento e utiliza apenas as atualizações compartilhadas ao longo das rodadas de agregação. Seja $\Delta\theta_t$ a atualização observada na rodada t, obtida diretamente a partir dos gradientes locais em protocolos descentralizados, nos quais ocorre troca explícita de gradientes, ou indiretamente como diferença entre instâncias consecutivas do modelo global; o ataque consiste em treinar um classificador auxiliar a partir de atualizações geradas por dados auxiliares rotulados segundo a propriedade privada p. Mais especificamente, o atacante utiliza um conjunto auxiliar do mesmo domínio, ou suficientemente próximo, para formar lotes com e sem a propriedade de interesse, obtendo pares $(\Delta\theta(b), a(b))$, em que $\Delta\theta(b)$ representa a atualização produzida pelo lote b e $a(b) \in \{0,1\}$ indica a presença ou ausência da propriedade nesse lote, que são então usados para treinar um classificador

$$A_{\text{prop}} : \Delta\theta \mapsto \{0,1\}, \tag{9}$$

capaz de estimar se a atualização observada no protocolo de Aprendizado Federado foi gerada por dados que contêm a propriedade privada [Melis et al., 2019].

Existe também uma variante ativa em que o participante adversário modifica seu próprio objetivo de treinamento para induzir o modelo conjunto a representar de forma mais nítida a propriedade privada de interesse [Melis et al., 2019]. A ideia é acoplar à tarefa principal uma tarefa auxiliar de predição de propriedade, substituindo a otimização puramente voltada à tarefa de classificação original por um objetivo multitarefa da forma

$$\mathscr{L}_{\text{adv}} = \mathscr{L}_{\text{main}} + \lambda\, \mathscr{L}_{\text{prop}}, \tag{10}$$

em que $\mathscr{L}_{\text{main}}$ corresponde à perda da tarefa legítima, $\mathscr{L}_{\text{prop}}$ força a rede a separar internamente exemplos que satisfazem ou não p, e $\lambda > 0$ controla a intensidade da interferência adversarial. Nesse cenário, o adversário não apenas observa o vazamento, mas amplifica a informatividade das atualizações futuras ao induzir o modelo global a aprender a propriedade p nas representações intermediárias, transformando o protocolo de Aprendizado Federado em um mecanismo de extração orientada de informação sem exigir acesso direto aos dados privados [Melis et al., 2019].

Os **ataques de inferência de origem** têm como objetivo associar uma informação privada a uma fonte específica dentro do Aprendizado Federado. Essa origem pode assumir diferentes formas, como o usuário responsável por um registro, o cliente detentor de um rótulo privado ou a pessoa à qual um padrão reconstruído pode ser atribuído [Fu et al., 2022]. Seja $\mathscr{C} = \{1, \ldots, K\}$ o conjunto de clientes participantes; dado um registro x,

um conjunto de características observáveis $\psi(x)$ e um mecanismo atacante A, o problema pode ser formulado como uma tarefa de atribuição

$$A(\psi(x)) \to \hat{c} \in \mathscr{C}. \tag{11}$$

Uma instância desse tipo de inferência aparece em Aprendizado Federado vertical, no qual diferentes participantes compartilham o espaço de amostras, mas não o espaço de atributos. Fu *et al.* analisam um cenário em que apenas um participante possui os rótulos y, enquanto os demais detêm subconjuntos de atributos $x^{(1)}, \ldots, x^{(K)}$ [Fu et al., 2022]; o objetivo do atacante deixa de ser decidir se uma amostra participou do treinamento ou reconstruir diretamente seu conteúdo, passando a ser a recuperação do rótulo privado mantido por outro participante. Assim, dada uma amostra x e o modelo federado treinado, o adversário busca estimar

$$\hat{y} = \arg\max_{y} P(y \mid x^{(a)}, \mathscr{I}), \tag{12}$$

em que $x^{(a)}$ representa os atributos observados pelo participante adversário e $\mathscr{I}$ agrega as informações disponíveis durante o treinamento, como gradientes recebidos e o comportamento do modelo de base (*bottom model*). No Aprendizado Federado vertical, cada participante utiliza um modelo de base para transformar seus atributos locais em uma representação intermediária, que é posteriormente combinada pelo modelo superior (*top model*), responsável por gerar a predição global, de modo que o modelo de base controlado pelo participante sem rótulos pode aprender representações suficientemente alinhadas com y para servir como instrumento de inferência [Fu et al., 2022].

Hu *et al.* apresentam uma formulação mais direta de inferência de origem [Hu et al., 2023]. Nesse caso, o objetivo não é apenas descobrir se um registro é membro do conjunto de treinamento global, mas inferir qual cliente possui esse registro, tratando a inferência de origem como um problema de atribuição entre os clientes da federação. O adversário observa, para um mesmo registro, como os diferentes modelos locais respondem em termos de erro preditivo e utiliza essa informação como critério de decisão, sob a hipótese de que o cliente que efetivamente utilizou esse registro em seu treinamento local tende a possuir um modelo mais ajustado à amostra, refletindo menor perda em comparação com os demais [Hu et al., 2023]. Essa ideia é adaptada a três protocolos de agregação distintos, *Federated Stochastic Gradient Descent* (FedSGD), FedAvg e *Federated Model Distillation* (FedMD), para demonstrar que a inferência de origem persiste independentemente de a federação compartilhar gradientes, parâmetros ou predições sobre dados públicos [Hu et al., 2023].

Os **ataques de reconstrução de conteúdo** têm como objetivo recuperar, aproximar ou sintetizar o próprio dado privado utilizado no treinamento. A reconstrução de conteúdo procura obter uma representação explícita da entrada privada, como uma imagem, um texto ou um protótipo semântico associado aos dados do participante. Seja $\mathscr{I}$ o conjunto de observáveis disponíveis ao adversário, como gradientes e parâmetros do modelo global, o objetivo do ataque pode ser escrito como a construção de um estimador $\hat{x} = A(\mathscr{I})$, tal que $\hat{x}$ preserve, na maior medida possível, a similaridade semântica ou estrutural com x. Em termos de otimização, isso equivale a buscar uma reconstrução

que minimize uma distância $d(\hat{x},x)$, ainda que x não seja diretamente observável pelo adversário [Zhu et al., 2019].

Zhu *et al.* formulam o problema e introduzem o ataque *Deep Leakage from Gradients* (DLG) [Zhu et al., 2019]. O cenário considerado é o de aprendizado distribuído ou colaborativo em que o adversário observa os gradientes compartilhados por outro cliente. Seja (x,y) um par privado de entrada e rótulo, $F(\cdot;W)$ um modelo diferenciável e o gradiente verdadeiro dado por

$$\nabla W = \frac{\partial \mathscr{L}(F(x;W),y)}{\partial W}, \tag{13}$$

o ataque parte de uma entrada fictícia x' e de um rótulo fictício y', ambos inicializados aleatoriamente, e computa os gradientes correspondentes

$$\nabla W' = \frac{\partial \mathscr{L}(F(x';W),y')}{\partial W}. \tag{14}$$

Em seguida, resolve-se o problema de otimização

$$\min_{x',y'} \left\| \nabla W' - \nabla W \right\|_2^2, \tag{15}$$

em que x' e y' são atualizados para minimizar a discrepância entre os gradientes fictícios e os gradientes observados.

A intuição é que, se os gradientes calculados sobre (x',y') coincidirem com os gradientes originalmente compartilhados, então o par fictício se torna uma aproximação do par privado real. O resultado central de DLG é mostrar que esse procedimento pode recuperar imagens com elevada fidelidade pixel a pixel e textos, evidenciando que o gradiente compartilhado preserva informação suficiente para reconstrução explícita do conteúdo [Zhu et al., 2019].

Apesar de sua importância, DLG apresenta limitações práticas, especialmente no tratamento do rótulo. Zhao *et al.* mostram que, no DLG original, o atacante precisa ajustar simultaneamente uma entrada fictícia x' e um rótulo fictício y' para que os gradientes gerados por esse par coincidam com os gradientes observados [Zhao et al., 2020a]. Como o rótulo verdadeiro não é previamente conhecido, ele também é tratado como variável de otimização, o que torna o problema mais instável e dificulta a convergência para a reconstrução correta. Para contornar esse problema, os autores propõem o *Improved Deep Leakage from Gradients* (iDLG), cuja principal contribuição é revelar que, em tarefas de classificação com perda de entropia cruzada e rótulos *one-hot*, o rótulo verdadeiro pode ser inferido analiticamente a partir dos sinais do gradiente da última camada totalmente conectada [Zhao et al., 2020a].

Seja g_i o gradiente da perda em relação ao *logit* da classe i, dado por $g_i = \frac{\partial \mathscr{L}}{\partial y_i}$. Os autores demonstram que, para a classe correta c, vale $g_c \in (-1,0)$, enquanto, para as demais classes $i \neq c$, $g_i \in (0,1)$. Quando os gradientes em relação aos *logits* não estão diretamente disponíveis, a mesma estrutura de sinal pode ser observada no gradiente dos parâmetros da última camada, permitindo identificar o rótulo verdadeiro antes da

reconstrução da entrada. Com isso, o problema de otimização deixa de envolver simultaneamente x' e y', reduzindo-se à

$$\min_{x'} \left\| \nabla W'(x', c) - \nabla W \right\|_2^2, \tag{16}$$

em que c já foi inferido. Essa modificação torna o processo mais estável, acelera a convergência e melhora a fidelidade da reconstrução, evidenciando que parte substancial do conteúdo supervisionado já vaza diretamente na estrutura do gradiente [Zhao et al., 2020a].

A reconstrução de conteúdo também pode ser conduzida por mecanismos generativos. Hitaj *et al.* mostram que, em aprendizado colaborativo profundo, um participante adversário pode utilizar Redes Generativas Adversariais (*Generative Adversarial Networks* – GANs) para sintetizar amostras prototípicas associadas aos dados privados de outro participante [Hitaj et al., 2017]. O ataque parte da observação de que, durante o treinamento, os parâmetros do modelo global são continuamente influenciados pelos dados locais dos participantes honestos. O adversário explora esse processo em tempo real, utilizando o modelo global como discriminador em uma GAN. Seja $G(z)$ o gerador, em que z é um vetor de ruído, e seja $D(\cdot)$ o discriminador fornecido pelo próprio treinamento colaborativo. O treinamento adversarial segue a lógica usual,

$$\min_G \max_D \; \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))], \tag{17}$$

mas, diferentemente do uso benigno de GANs para modelagem de distribuição, nesse caso o gerador é empregado de forma maliciosa para aproximar a distribuição dos dados privados do participante alvo [Hitaj et al., 2017]. É importante ressaltar que, quando a GAN é treinada, a esperança é aproximada pela média no *mini-batch*,

$$\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] \approx \frac{1}{m} \sum_{i=1}^{m} \log D(x_i). \tag{18}$$

O ataque é particularmente relevante porque não visa necessariamente reconstruir uma amostra exata do conjunto local, como em DLG e iDLG, mas produzir conteúdo sintético suficientemente semelhante para revelar informação sensível sobre a distribuição privada aprendida durante o treinamento colaborativo [Hitaj et al., 2017].

Do ponto de vista operacional, os ataques DLG e iDLG são ataques de reconstrução por otimização guiada por gradientes. O objeto central observado pelo adversário é o gradiente compartilhado e a reconstrução decorre do ajuste iterativo de uma entrada fictícia até que ela induza o mesmo padrão diferencial do exemplo real [Zhu et al., 2019, Zhao et al., 2020a]. Já na reconstrução com GAN, o adversário utiliza a evolução do modelo colaborativo para treinar um gerador capaz de produzir amostras sintéticas da distribuição privada do alvo [Hitaj et al., 2017]. Em ambos os casos, a hipótese subjacente é a mesma de que o processo de treinamento colaborativo não transmite apenas conhecimento abstrato sobre a tarefa, mas preserva informação suficiente para aproximar o conteúdo privado que o originou.

1.4. Estratégias de Mitigação

O paradigma do Aprendizado Federado é ciente de privacidade para o treinamento de modelos, já que não há necessidade de compartilhar dados brutos e os participantes

podem decidir livremente ingressar ou deixar a federação, mas não fornece garantias suficientes de privacidade e robustez. Os mecanismos atuais tendem a serem vulneráveis, principalmente, (i) ao servidor curioso ou malicioso que busca inferir informações sensíveis dos clientes a partir das atualizações do modelo e (ii) a participantes adversários que têm como objetivo envenenar o modelo global [Lyu et al., 2022].

1.4.1. Técnicas de Preservação de Privacidade

Para ataques de inferência e de inversão de modelo, a privacidade diferencial surge como uma abordagem promissora. A privacidade diferencial define um arcabouço matemático que garante privacidade aos usuários por meio da inserção de ruído aleatório durante a análise ou o compartilhamento de dados. No contexto de Aprendizado Federado, destacam-se três tipos de soluções com privacidade diferencial: (i) a privacidade diferencial centralizada (*Centralized Differential Privacy* – CDP) [McMahan et al., 2018]; (ii) a privacidade diferencial local (*Local Differential Privacy* – LDP) [Wang et al., 2019, Zhao et al., 2020c, Truex et al., 2020, Sun et al., 2021]; e (iii) a privacidade diferencial distribuída (*Distributed Differential Privacy* – DDP) [Agarwal et al., 2018, Truex et al., 2019, Lyu, 2020]. Nessas variantes, a garantia de privacidade recai sobre valores agregados ou valores locais, a depender de quem é responsável por adicionar o ruído.

Os algoritmos CDP foram inicialmente descritos para aprendizado de máquina centralizado, no qual todos os dados se encontram em um único repositório. Ao adaptá-los ao Aprendizado Federado, assume-se que o servidor central é confiável e fica encarregado de adicionar o ruído aleatório após agregar os parâmetros recebidos dos clientes, utilizando, por exemplo, distribuições laplaciana ou gaussiana [Dwork e Roth, 2014]. Essa suposição de confiança, porém, nem sempre é realista e introduz um ponto único de falha para vazamentos de dados dos clientes, motivo pelo qual, em grande parte das aplicações de FL, recorrem-se a algoritmos baseados em LDP e DDP.

Nos algoritmos LDP, a responsabilidade por inserir ruído recai sobre os próprios clientes, que perturbam seus dados privados antes mesmo do treinamento local e do envio de parâmetros ao servidor. Para gerar esse ruído, o cliente pode empregar as mesmas distribuições laplaciana ou gaussiana usadas em CDP ou ainda modificar seus dados de modo que o servidor não consiga distinguir se os valores agregados refletem atualizações reais ou valores previamente determinados. Como o cliente desconhece o impacto exato de seus dados no modelo global, ele tende a inserir quantidades maiores de ruído para garantir privacidade diferencial, o que afeta a convergência do modelo, especialmente em cenários com centenas ou milhares de clientes, nos quais o erro aditivo em LDP é bem maior do que em CDP. Como consequência, muitas técnicas de LDP são mais adequadas a modelos com poucos parâmetros e a conjuntos de dados simples.

Como algoritmos LDP puros prejudicam significativamente a convergência e algoritmos CDP dependem de uma hipótese forte de confiança no servidor, surgiram os algoritmos DDP, que combinam inserção de ruído em parte dos dados dos clientes com funções criptográficas no servidor. Essa combinação restringe o conjunto de distribuições adequadas, privilegiando distribuições mais estáveis, como a gaussiana ou a binomial, que permitem a cada cliente adicionar ruídos de menor magnitude do que em LDP, ainda que a privacidade diferencial só seja estritamente satisfeita após a agregação dos valores

de todos os clientes [Agarwal et al., 2018]. Devido a essa característica, algoritmos DDP são tipicamente associados a mecanismos de computação multipartidária segura (*Secure Multi-Party Computation* – SMC), para reforçar as garantias de segurança sobre os dados dos participantes [Agarwal et al., 2018].

Outra abordagem é a criptografia homomórfica, técnica na qual computações podem ser realizadas diretamente sobre dados cifrados, sem necessidade de descriptografá-los [Lim et al., 2020, de Assis et al., 2023, de Assis et al., 2024]. Quando os dados são finalmente descriptografados, o resultado coincide com aquele que seria obtido ao realizar a mesma computação em texto plano, o que permite executar operações de agregação diretamente sobre dados cifrados e impede que um servidor não confiável acesse informações sensíveis e realize ataques de inferência. Embora ofereça um alto nível de proteção contra ataques de inferência por parte de servidores curiosos, o custo computacional dessa abordagem é elevado e pode ser inviável em dispositivos com recursos limitados, como aqueles comumente empregados em cenários de IoT.

1.4.2. Agregação Robusta

Técnicas de agregação robusta procuram combinar as atualizações de modelo dos participantes por meio de algoritmos resilientes a valores atípicos, ruído e manipulação adversária [Pillutla et al., 2022]. Em geral, esses métodos favorecem atualizações consistentes com a maioria e reduzem a influência de clientes maliciosos, sendo úteis contra ataques Bizantinos e ataques de envenenamento de modelo [Cunha Neto et al., 2023].

Ataques de envenenamento não-direcionados, em especial ataques Bizantinos, têm como objetivo comprometer a convergência do modelo global por meio da inserção de atualizações maliciosas. A defesa contra esses ataques é desafiadora porque o servidor observa apenas gradientes ou parâmetros dos modelos locais, sem acesso direto aos dados dos participantes, o que torna muitas defesas do aprendizado centralizado ineficazes. Como não é possível confiar plenamente nos participantes, o servidor precisa detectar atualizações anômalas ou reduzir a influência de valores atípicos durante a agregação, de modo que o modelo ainda consiga convergir mesmo quando uma fração dos clientes se comporta de forma maliciosa.

Uma primeira classe de defesas consiste em identificar e filtrar participantes maliciosos antes da agregação, como nos métodos AUROR [Shen et al., 2016], FoolsGold [Fung et al., 2018], FLTrust [Cao et al., 2020], Zeno++ [Xie et al., 2020] e COSDefense [Yaldiz et al., 2023]. O AUROR utiliza *k-means* para agrupar participantes com distribuições estatísticas similares nas características mais relevantes do modelo e descarta as atualizações fora desse agrupamento. O FoolsGold descarta atualizações de participantes que apresentam gradientes excessivamente semelhantes ao longo do tempo. O FLTrust explora um conjunto de dados confiável para calcular um "gradiente de referência" e remove participantes cujas atualizações se desviam demais desse gradiente segundo a similaridade de cosseno. O Zeno++ avalia as contribuições com base na variação da função de custo em um conjunto de validação, descartando atualizações que não melhoram ou degradam esse valor. O COSDefense analisa a similaridade de cosseno entre gradientes para identificar participantes alinhados em uma direção comum e descartar gradientes inconsistentes com essa direção.

Uma segunda classe de defesas modifica diretamente a regra de agregação para torná-la menos sensível a valores atípicos ou contribuições maliciosas, como em Krum [Blanchard et al., 2017], *Trimmed Mean* [Yin et al., 2018], Bulyan [Guerraoui et al., 2018] e *Robust Federated Aggregation* (RFA) [Pillutla et al., 2022]. O Krum calcula, para cada participante, a soma das distâncias euclidianas até seus vizinhos mais próximos e seleciona a atualização que é mais consistente com a maioria. O *Trimmed Mean* remove, coordenada a coordenada, os valores máximo e mínimo e calcula a média dos valores remanescentes. O Bulyan combina as ideias de Krum e *Trimmed Mean*: primeiro escolhe um subconjunto de atualizações mais confiáveis usando a regra de Krum e, em seguida, aplica o *Trimmed Mean* nesse subconjunto. O RFA utiliza a mediana geométrica em vez da média aritmética, buscando o ponto que minimiza a soma das distâncias a todas as atualizações e reduzindo, assim, o impacto de contribuições maliciosas.

Apesar de mitigarem ataques não-direcionados até certo ponto, essas técnicas de defesa não são suficientes contra ataques direcionados, como ataques *backdoor*, o que motiva o desenvolvimento de estratégias complementares, além da agregação robusta.

Diferentemente dos ataques não-direcionados, que buscam desestabilizar a convergência do modelo global, **ataques direcionados** (ataques de envenenamento *backdoor*) têm como objetivo manipular seletivamente o comportamento do modelo. Nesses cenários, as atualizações maliciosas são projetadas para se misturarem às atualizações benignas, de forma a permanecerem compatíveis com as estatísticas globais e, assim, burlar métodos agregação robusta e simples mecanismos de filtragem de anomalias. Por conta dessa natureza mais sutil, defesas contra ataques direcionados são tipicamente organizadas em duas famílias principais: (i) métodos de detecção, voltados a identificar a presença de um *backdoor*, e (ii) métodos de eliminação, voltados a remover ou mitigar o efeito do envenenamento já incorporado ao modelo [Li et al., 2022b].

Métodos de detecção exploram estatísticas das funções de ativação e propriedades estruturais do modelo para determinar se este foi envenenado por um ataque *backdoor*. A ideia central é que dados envenenados, ao serem incorporados ao treinamento, deixam traços característicos na representação latente aprendida, que podem ser diferenciados dos padrões produzidos por dados limpos. Tran *et al.* introduzem a noção de "assinatura espectral", segundo a qual amostras envenenadas produzem uma anomalia espectral na matriz de características internas do modelo [Tran et al., 2018]. Após o treinamento, constrói-se a matriz de covariância das representações e aplica-se *Single Value Decomposition* (SVD). As amostras envenenadas tendem a se projetar ao longo da direção de maior variância, permitindo identificá-las a partir dos componentes principais dominantes. Chen *et al.* propõem o *Activation Clustering* que parte da hipótese de que amostras legítimas e amostras envenenadas não geram o mesmo padrão de ativações nas camadas escondidas [Chen et al., 2018]. Nesse caso, coletam-se as saídas das funções de ativação para uma dada classe, aplica-se um algoritmo de agrupamento e verifica-se se as amostras dessa classe se organizam em dois grupos bem separados, dos quais um é associado às instâncias envenenadas.

Métodos de eliminação assumem que o ataque já foi bem-sucedido e visam remover o *backdoor* do modelo global treinado, reduzindo ou anulando o comportamento malicioso sem destruir completamente a capacidade do modelo na tarefa principal. Liu *et*

al. propõem o *Fine-Pruning*, que explora o fato de que o *backdoor* tende a ser codificado em neurônios pouco ativados por amostras legítimas, mas fortemente ativados na presença do gatilho [Liu et al., 2018]. A técnica mede a ativação média de cada neurônio sob dados limpos, poda aqueles com baixa ativação e realiza um ajuste fino (*fine-tuning*) com os neurônios restantes, buscando preservar o desempenho legítimo enquanto remove a lógica associada ao *backdoor*. Li *et al.* introduzem o *Neural Attention Distillation* (NAD), que utiliza um modelo instrutor, treinado em dados confiáveis, para guiar a recuperação de um modelo aprendiz envenenado [Li et al., 2021b]. O método alinha os mapas de atenção das camadas escondidas do aprendiz com os do instrutor, incentivando o modelo comprometido a concentrar-se novamente em padrões semânticos legítimos e a deixar de depender dos padrões de acionamento do *backdoor*.

Outra linha é o *Mode Connectivity Repair* (MCR), proposto por Zhao *et al.*, que se apoia em propriedades geométricas da paisagem de perda de redes neurais [Zhao et al., 2020b]. A premissa é que um modelo treinado apenas com dados legítimos e um modelo envenenado correspondem a dois pontos no espaço de parâmetros que podem ser conectados por um caminho de baixa perda. Ao construir explicitamente esse caminho e otimizar o valor esperado da perda ao longo da curva, é possível deslocar o modelo envenenado em direção a uma região "saudável" da paisagem de perda, reparando o comportamento adversarial sem reiniciar o treinamento do zero. Finalmente, o *Anti-Backdoor Learning* (ABL), também proposto por Li *et al.*, explora o fato de que padrões associados ao *backdoor* são aprendidos mais rapidamente e normalmente concentrados em uma classe específica [Li et al., 2021a]. O ABL aplica gradiente ascendente nas amostra, monitorando a evolução da perda para distinguir exemplos que apresentam rápido decaimento (típico de instâncias envenenadas) daqueles com dinâmica mais gradual (típico de exemplos legítimos). Uma vez separadas, aplica-se gradiente ascendente sobre o conjunto de amostras classificadas como envenenadas, forçando o modelo a desaprender o comportamento de *backdoor* associado a essas instâncias.

Em conjunto, essas estratégias mostram que a defesa contra ataques direcionados exige ir além da mera robustez de agregação: é necessário analisar representações internas, dinâmica de treinamento e geometria da superfície de perda para identificar e remover, de forma seletiva, os comportamentos maliciosos inseridos no modelo. A Tabela 1.1 sintetiza os principais tipos de ataques em Aprendizado Federado e as respectivas estratégias de mitigação discutidas nas seções anteriores.

1.5. Atividade Prática

Essa seção descreve as atividades práticas, a implementação de ataques relevantes em Aprendizado Federado utilizando o arcabouço Flower, na versão 1.25.0, em um ambiente experimental controlado. O objetivo é permitir que os participantes compreendam, na prática, como diferentes classes de ataques podem ser implementadas, executadas e avaliadas em um cenário federado.

A atividade foi estruturada em torno dos dois tipos de ataques ao FL, os ataques ao desempenho do modelo e os ataques à privacidade. No primeiro grupo, são implementados três ataques bizantinos, *A Little Is Enough* (ALIE), *Sign Flipping* e injeção de ruído gaussiano, com o propósito de mostrar como clientes maliciosos podem manipular

Tabela 1.1. Tipos de ataques em Aprendizado Federado e estratégias de mitigação associadas.

Tipo de ataque	Subtipo	Mecanismo de ação	Defesa principal	Mecanismo de defesa
Envenenamento do Modelo Ataques Bizantinos	Envenenamento de dados	Cliente altera rótulos, atributos ou insere exemplos sintéticos para enviesar gradientes locais e degradar o modelo global.	Agregação robusta (Krum, *Trimmed Mean*, Bulyan, RFA) [Blanchard et al., 2017, Yin et al., 2018, Guerraoui et al., 2018, Pillutla et al., 2022]	Ajusta a regra de agregação para reduzir o peso de valores atípicos e atualizações maliciosas.
	Envenenamento de modelo	Cliente manipula diretamente direção e norma da atualização enviada ao servidor para desviar a trajetória de otimização.	Filtragem de clientes (AUROR, FoolsGold, FLTrust, Zeno++, COSDefense) [Shen et al., 2016, Fung et al., 2018, Cao et al., 2020, Xie et al., 2020, Yaldiz et al., 2023]	Detecta e descarta atualizações suspeitas com base em similaridade estatística, gradientes ou impacto em validação.
	Bizantino descentralizado	Clientes desonestos difundem modelos locais corrompidos na vizinhança em topologias sem servidor confiável.	Agregação robusta descentralizada [Fang et al., 2024]	Usa regras robustas de consenso entre vizinhos para limitar a propagação de atualizações anômalas.
Ataques à privacidade	Inferência de participação	Adversário estima se uma amostra específica participou do treinamento a partir de saídas, gradientes ou diferenças de parâmetros.	Privacidade diferencial (CDP, LDP, DDP) [McMahan et al., 2018, Wang et al., 2019, Agarwal et al., 2018, Lyu, 2020]	Injeta ruído calibrado em parâmetros ou atualizações para limitar a influência de cada exemplo no modelo.
	Inferência de atributos	Adversário infere propriedades latentes ou sensíveis dos dados com base em atualizações e representações internas.	LDP/DDP com ruído local e distribuído [Wang et al., 2019, Agarwal et al., 2018]	Perturba gradientes locais para mascarar correlações entre atributos privados e parâmetros.
	Inferência de origem	Adversário vincula registros ou rótulos a um cliente específico usando respostas de modelos locais.	Aprendizado Federado vertical protegido e SMC [Fu et al., 2022, Agarwal et al., 2018]	Emprega computação segura e separação de atributos/rótulos para reduzir o vazamento de autoria dos dados.
	Reconstrução de conteúdo (DLG, iDLG, GAN)	Adversário reconstrói exemplos privados (imagens, textos) a partir de gradientes ou da evolução do modelo.	Privacidade diferencial e criptografia homomórfica [Zhu et al., 2019, Zhao et al., 2020a, Lim et al., 2020, de Assis et al., 2023, de Assis et al., 2024]	Adiciona ruído e realiza agregação sobre dados cifrados para limitar a informação útil à reconstrução.
Ataques direcionados *backdoor*	Backdoor por dados com gatilho	Cliente insere exemplos com padrão de gatilho e rótulo alvo para ativar saídas maliciosas sob condições específicas.	Detecção por representação (assinatura espectral, agrupamento de ativação) [Tran et al., 2018, Chen et al., 2018]	Analisa representações internas para identificar subgrupos de exemplos com padrões latentes discrepantes associados ao *backdoor*.
	Substituição de modelo (*model replacement*)	Cliente reescala sua atualização para aproximar o modelo global de um modelo local envenenado, mantendo boa acurácia padrão.	Agregação robusta e limitação de norma [Wang et al., 2020, Pillutla et al., 2022]	Impõe limites de norma e regras robustas para evitar que uma única atualização domine o modelo global.
	Backdoor por casos raros	Cliente usa exemplos raros, porém legítimos, associados a um rótulo alvo para esconder o gatilho em regiões pouco amostradas.	Fine-Pruning, NAD, MCR, ABL [Liu et al., 2018, Li et al., 2021b, Zhao et al., 2020b, Li et al., 2021a]	Remove ou enfraquece o *backdoor* podando neurônios, realinhando atenção, explorando conectividade de modos e desaprendendo padrões maliciosos.
Ataques ao protocolo	Servidor curioso	Servidor tenta inferir dados sensíveis a partir das atualizações recebidas ou quebra suposições de honestidade.	Criptografia homomórfica e SMC [Lim et al., 2020, de Assis et al., 2023, de Assis et al., 2024, Agarwal et al., 2018]	Permite agregação sobre dados cifrados ou via computação segura, ocultando atualizações em texto claro.
	Exploração de heterogeneidade / *distributional shift*	Adversário explora a variabilidade natural dos dados e mudanças de distribuição para camuflar atualizações maliciosas.	Agregação robusta com validação de referência [Zhang et al., 2025, Li et al., 2023]	Combina regras robustas com validação em dados de referência para separar heterogeneidade benigna de desvios adversariais.

atualizações locais para degradar a convergência e o desempenho do modelo global. No segundo grupo, são implementados dois ataques voltados à privacidade, *Deep Leakage from Gradients* (DLG) e reconstrução baseada em GAN, evidenciando que o compartilhamento de gradientes ou atualizações pode expor informações sensíveis sobre clientes.

1.5.1. Configuração do Cenário do Aprendizado Federado no Flower

O Flower é um arcabouço de Aprendizado Federado que organiza a execução em dois componentes centrais: o `ServerApp`, responsável pela estratégia e pela orquestração do treinamento, e o `ClientApp`, responsável pelo treinamento e pela avaliação local em cada cliente. A comunicação entre servidor e clientes é realizada por meio de mensagens que transportam um `RecordDict`, estrutura de dados que pode conter diferentes tipos de registros, como `ArrayRecord`, `MetricRecord` e `ConfigRecord`. O `ArrayRecord` é utilizado para armazenar estruturas numéricas serializadas, tipicamente associadas aos parâmetros do modelo ou a outros arranjos numéricos relevantes para o treinamento. Na prática, ele é usado para encapsular os parâmetros do modelo global enviados pelo servidor aos clientes e, posteriormente, os parâmetros ou atualizações locais devolvidos pelos clientes ao servidor.

Em aplicações com PyTorch, o `ArrayRecord` pode ser construído diretamente a partir de um `state_dict` e convertido novamente para esse formato no momento da execução local. Por essa razão, o `ArrayRecord` constitui o principal mecanismo para transportar o estado do modelo ao longo das rodadas de comunicação. O `MetricRecord`, por sua vez, é utilizado para armazenar métricas numéricas produzidas durante o treinamento ou a avaliação. Essas métricas podem incluir, por exemplo, perda do treinamento, acurácia, número de exemplos processados e outros valores de interesse para monitoramento ou agregação. Já o `ConfigRecord` é utilizado para transportar parâmetros de configuração entre servidor e clientes.

Para iniciar um novo ambiente no Flower, utiliza-se o comando `flwr new`, que cria automaticamente um esqueleto inicial do projeto. Uma instalação típica do ambiente de simulação pode ser iniciada com os comandos do Código 1.1. O projeto gerado já organiza a aplicação em arquivos como `client_app.py`, `server_app.py` e `task.py`. O arquivo `client_app.py` contém a lógica executada pelos clientes, isto é, o treinamento e a avaliação local. O arquivo `server_app.py` contém a lógica do servidor, incluindo a definição do protocolo de agregação, chamado de *Strategy* pelo Flower, e a coordenação das rodadas de treinamento. Já o arquivo `task.py` concentra elementos compartilhados pela aplicação, como definição do modelo, carregamento do conjunto de dados, particionamento dos dados e funções auxiliares de treino e teste.

Os arquivos gerados constituem um exemplo simples de Aprendizado Federado que pode ser facilmente modificado para cenários mais complexos alterando poucas linhas de código. Para modificar o conjunto de dados, basta alterar a parte de carregamento e particionamento dos dados no arquivo `task.py`, onde são definidos o conjunto de dados utilizado, o pré-processamento aplicado e o particionador responsável por distribuir as amostras entre os clientes. Para mudar a distribuição dos dados dos clientes, basta substituir ou reconfigurar o particionador utilizado no `task.py`. Em particular, pode-se usar o `IidPartitioner` para simular um cenário IID ou o `DirichletPartitioner`

para simular um cenário *non-IID*, ajustando parâmetros como o número de partições e o valor de `alpha`, que controla o grau de heterogeneidade entre os clientes. Para mudar as configurações da simulação, como número de clientes, número de rodadas, épocas locais, taxa de aprendizado, tamanho do lote e outros hiperparâmetros globais, basta modificar o arquivo de configuração `pyproject.toml`. Nesse arquivo, as opções da simulação são definidas na seção de configuração da aplicação Flower e ficam disponíveis durante a execução por meio de `context.run_config` e `context.node_config`.

Após a criação do projeto, a instalação local pode ser concluída com `pip install -e .` e a execução da simulação pode ser realizada por meio de `flwr run .`.

```
pip install -U "flwr[simulation]"
flwr new fl-cifar100-attacks --framework pytorch --username <
    seu_usuario>
cd fl-cifar100-attacks
pip install -e .
flwr run .
```

Código Fonte 1.1. Inicialização de um novo projeto Flower

Com essa configuração, estabelece-se uma base experimental apropriada para a implementação dos ataques. O Flower fornece a infraestrutura de comunicação e orquestração e os particionamentos *IID* e *non-IID* permitem investigar o comportamento dos ataques sob diferentes graus de heterogeneidade estatística. A partir desse cenário, torna-se possível introduzir, de forma controlada, tanto ataques bizantinos voltados à degradação do desempenho do modelo global quanto ataques de privacidade voltados à extração ou reconstrução de informações sensíveis.

1.5.2. Modelo do Atacante

Na atividade prática, os ataques são organizados segundo dois modelos de adversário distintos. Para os ataques ao desempenho do modelo global, adota-se um modelo em que o atacante atua como um cliente participante do treinamento de FL. Para os ataques à privacidade, adotam-se dois modelos de adversário complementares. No caso do DLG, o adversário corresponde ao próprio servidor agregador, assumido como *honesto-mas-curioso*. Já no caso da reconstrução com GAN, o adversário é modelado como um cliente participante curioso, que segue o protocolo federado, mas explora as informações disponíveis durante o treinamento para inferir conteúdo privado de outro participante. A adoção desses dois modelos distintos tem como objetivo enriquecer a demonstração prática, mostrando que os riscos à privacidade em Aprendizado Federado podem emergir tanto do lado do servidor quanto do lado dos próprios clientes participantes.

No caso dos ataques bizantinos, o atacante é modelado como um cliente participante do sistema de Aprendizado Federado, isto é, como um cliente que pode ou não ser selecionado para participar das rodadas de treinamento, com acesso normal ao modelo global enviado pelo servidor e com capacidade de retornar uma atualização local manipulada se selecionado. Esse atacante segue o fluxo esperado do protocolo de agregação no sentido de receber o modelo, executar localmente seu procedimento e responder ao servidor no momento previsto. Entretanto, em vez de enviar uma contribuição compatível com o treinamento benigno, esse cliente envia uma atualização adulterada com o objetivo

de degradar a convergência, reduzir o desempenho final ou desestabilizar o treinamento global. Esse modelo é consistente com a caracterização de ataques bizantinos discutida no texto, em que clientes maliciosos se desviam do comportamento esperado e enviam atualizações adversariais ao processo de agregação. No contexto deste trabalho, os ataques ALIE, *Sign Flipping* e injeção de ruído gaussiano seguem exatamente essa lógica. Em todos esses casos, o adversário não precisa controlar o servidor nem comprometer toda a federação. Basta que um ou mais clientes participantes estejam sob controle malicioso para que as atualizações enviadas ao agregador passem a induzir desvios na trajetória de otimização do modelo global.

Já nos ataques à privacidade, o modelo adversarial adotado é diferente. Nesse caso, assume-se que o atacante pode ser o servidor agregador e que esse servidor opera sob o modelo *honesto-mas-curioso*, ou um participante curioso. Um servidor *honesto-mas-curioso* é uma entidade que executa corretamente o protocolo de FL, isto é, envia os parâmetros esperados aos clientes, recebe as respostas e realiza a agregação conforme definido, sem alterar explicitamente o fluxo da execução. No entanto, embora siga o protocolo de forma correta, esse servidor tenta extrair o máximo possível de informação sensível a partir dos dados que observa durante o treinamento, como gradientes, parâmetros locais, diferenças entre parâmetros ou outras atualizações recebidas dos clientes. Em outras palavras, trata-se de um adversário que é honesto do ponto de vista operacional, mas curioso do ponto de vista informacional. Esses modelos são amplamente utilizados na análise de privacidade em Aprendizado Federado porque refletem um cenário em que o principal risco não decorre da quebra explícita do protocolo, mas da possibilidade de inferência a partir das informações legitimamente acessíveis ao agregador [Lim et al., 2020, Yang et al., 2019]. Esses modelos são adequados para os ataques DLG e reconstrução com GAN considerados nesta atividade prática.

1.5.3. Implementação dos Ataques ao Desempenho do Modelo

Os ataques bizantinos são implementados no arquivo `client_app.py`. O código fonte encontra-se disponível no GitLab[7] A ideia central da implementação dessa simulação é que cada cliente recebe o modelo global enviado pelo servidor, realiza o treinamento local normalmente e, em seguida, caso seja identificado como malicioso, aplica uma transformação adversarial sobre os parâmetros do modelo antes de devolvê-los ao servidor. Com isso, o ataque não altera a estrutura do protocolo de comunicação do Flower, mas interfere diretamente no conteúdo da atualização enviada ao agregador.

Para esse fim, foram criadas três funções específicas em `client_app.py`: `apply_gaussian_noise_attack()`, `apply_alie_attack()` e `apply_sign_flip_attack()`. A função `apply_gaussian_noise_attack()` implementa o ataque por ruído gaussiano. Nesse ataque, o cliente malicioso adiciona perturbações aleatórias aos parâmetros do modelo treinado localmente. O objetivo é corromper a atualização local sem seguir uma direção de otimização consistente, inserindo ruído suficiente para degradar a qualidade da agregação global.

[7]Disponível em `https://gitlab.com/LCC/fl-performance-attack`. Acessado em 21/04/2026

```python
def apply_gaussian_noise_attack(model: torch.nn.Module, noise_std: float = 1.0) -> None:
    with torch.no_grad():
        for param in model.parameters():
            noise = torch.randn_like(param) * noise_std
            param.add_(noise)
```

Código Fonte 1.2. Implementação do ataque por ruído gaussiano

A função recebe como entrada o modelo local e um parâmetro `noise_std`, que controla o desvio padrão do ruído gaussiano. O bloco `with torch.no_grad()` é utilizado para garantir que essa modificação não seja registrada pelo mecanismo de diferenciação automática do PyTorch, já que o objetivo não é realizar retropropagação, mas apenas alterar diretamente os valores numéricos dos parâmetros. Em seguida, a função percorre todos os parâmetros do modelo. Para cada parâmetro, é gerado um tensor de ruído com a mesma dimensão do tensor original por meio de `torch.randn_like(param)`, sendo esse ruído escalado pelo fator `noise_std`. Por fim, o ruído é somado diretamente ao parâmetro com `param.add_(noise)`.

Do ponto de vista do ataque, essa implementação produz uma atualização local ruidosa, cuja direção não corresponde à direção benigna da otimização. Assim, quando essa atualização é agregada juntamente com atualizações benignas, ela pode deslocar o modelo global para distantes do mínimo global da função de perda, prejudicando a convergência e a acurácia final.

A função `apply_alie_attack()` implementa o ataque *A Little Is Enough* (ALIE). Diferentemente do ruído gaussiano, o ALIE procura construir uma atualização maliciosa com aparência estatisticamente plausível. A ideia é produzir uma atualização desviada, mas sem gerar uma anomalia extremamente evidente, dificultando a distinção entre comportamento benigno e comportamento malicioso.

```python
def apply_alie_attack(
    model: torch.nn.Module,
    global_state_dict: dict,
    num_clients: int,
    num_malicious: int,
    z_max: float | None = None,) -> None:

    from scipy.stats import norm

    n, m = num_clients, num_malicious

    if z_max is None:
        s = max(1, (n // 2 + 1) - m)
        p = (n - m - s) / max(n - m, 1)
        p = min(max(p, 1e-6), 1 - 1e-6)
        z_max = float(norm.ppf(p))

    with torch.no_grad():
        for (name, param), global_param in zip(
            model.named_parameters(), global_state_dict.values()
        ):
            global_param = global_param.to(param.device)
            mu = global_param.mean()
```

```
24              sigma = global_param.std()
25              low = mu - z_max * sigma
26              high = mu + z_max * sigma
27              param.data = torch.empty_like(param.data).uniform_(low.item
    (), high.item())
```

Código Fonte 1.3. Implementação do ataque ALIE

A função recebe o modelo local já treinado e o *state_dict* global, o número total de clientes, o número de clientes maliciosos e, opcionalmente, o parâmetro `z_max`. No Código 1.3 linhas 13 — 17, há uma estrutura para cálculo automático do `z_max` a partir de uma aproximação baseada na quantidade total de clientes lícitos e clientes maliciosos.

Após essa etapa, o código percorre simultaneamente os parâmetros do modelo local e os correspondentes do modelo global recebido do servidor. A seguir, a função calcula duas estatísticas escalares do parâmetro global, a média `mu = global_param.mean()` e o desvio-padrão `sigma = global_param.std()`. Como `global_param` é um tensor, essas operações retornam valores agregados sobre todos os elementos daquele tensor. Assim, a média representa o valor médio dos parâmetros da camada global, enquanto o desvio-padrão representa a dispersão global.

Com essas estatísticas, a função define um intervalo de amostragem $w_j \sim \mathcal{U}(\mu - z_{\max}\sigma,\ \mu + z_{\max}\sigma)$. Esse intervalo representa uma faixa centrada na média dos parâmetros globais e com largura proporcional ao desvio padrão escalado por `z_max`. Em seguida, o parâmetro local é completamente sobrescrito por novos valores gerados aleatoriamente de forma uniforme nesse intervalo e atribuído a cada parâmetro do modelo, por meio da instrução da linha 27.

A função `apply_sign_flip_attack()` implementa o ataque *Sign Flip*. Nesse caso, o objetivo é inverter o sinal de parâmetros do modelo local, direcionando a atualização na direção oposta àquela produzida pelo treinamento benigno. Trata-se de uma implementação direta de um ataque de envenenamento do modelo, no qual a contribuição enviada ao servidor passa a contrariar o processo de descida da função de perda.

```
1  def apply_sign_flip_attack(
2      model: torch.nn.Module,
3      flip_factor: float = -1.0,
4      top_fraction: float = 0.2,
5  ) -> None:
6      with torch.no_grad():
7          for param in model.parameters():
8              if top_fraction >= 1.0:
9                  param.mul_(flip_factor)
10             else:
11                 flat = param.data.view(-1)
12                 k = max(1, int(flat.numel() * top_fraction))
13                 _, top_indices = torch.topk(flat.abs(), k)
14                 flat[top_indices] *= flip_factor
```

Código Fonte 1.4. Implementação do ataque Sign Flip

A função recebe o modelo local, o fator de inversão `flip_factor` e a fração `top_fraction`. O valor padrão de `flip_factor` é -1.0, o que significa inverter o sinal dos parâmetros selecionados. Já `top_fraction` controla a fração de parâmetros

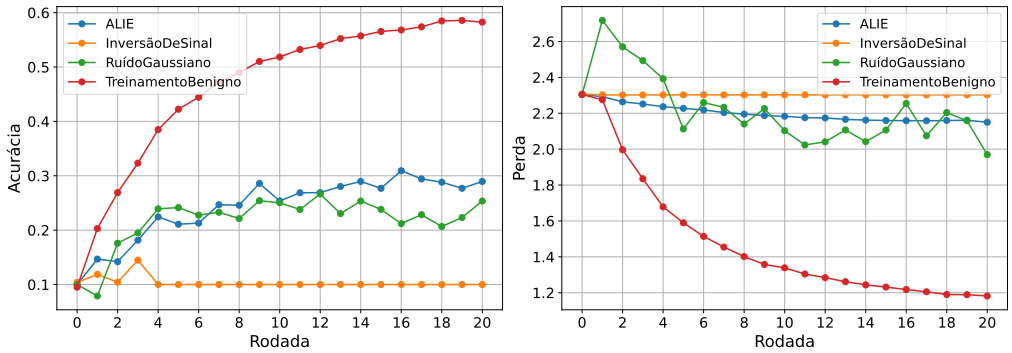


Figura 1.5. Evolução da acurácia e da perda ao longo das rodadas de agregação com protocolo FedAvg, comparando o treinamento benigno com os ataques. O gráfico da esquerda mostra a acurácia global por rodada, enquanto o gráfico da direita mostra a perda global correspondente.

que sofrerá inversão. Se `top_fraction >= 1.0`, todos os parâmetros da camada têm seus sinais invertidos. Caso contrário, a função seleciona apenas os parâmetros de maior magnitude absoluta dentro de cada tensor de parâmetros.

Essa seleção parcial é realizada achatando o tensor com `view(-1)`, calculando a quantidade k de elementos a serem alterados e aplicando `torch.topk(flat.abs(), k)` para obter os índices dos maiores valores absolutos. Em seguida, apenas esses elementos têm seus sinais invertidos. Essa implementação permite um ataque menos uniforme e potencialmente mais furtivo, concentrando a manipulação nos parâmetros de maior magnitude, que tendem a ter maior influência sobre a atualização final do modelo.

A aplicação efetiva dos ataques ocorre na função `train()`, que representa a rotina de treinamento local executada pelos clientes. Essa função não apenas realiza o treino benigno, mas também determina quais clientes serão maliciosos e o ataque.

Após o treinamento local, a função passa a decidir se aquele cliente será benigno ou malicioso. Para isso, são utilizados dois elementos do arquivo de configuração da aplicação. O primeiro é `attack-type`, que define qual ataque será aplicado, e o segundo é o parâmetro `malicious-fraction`, recuperados por meio de `context.run_config`. O parâmetro `malicious-fraction` indica a fração de clientes da federação que será tratada como maliciosa.

Essa lógica significa que os primeiros clientes, de acordo com o identificador de partição, são marcados como maliciosos. Por exemplo, se houver 10 clientes e `malicious-fraction = 0.2`, então `num_malicious = 2`, e os clientes com `partition_id` 0 e 1 serão considerados atacantes. Trata-se de uma escolha de implementação simples e determinística, suficiente para experimentos controlados.

Para tornar a simulação mais flexível, foram introduzidos o parâmetro `malicious-fraction` e `attack-type` no arquivo de configuração da aplicação (`pyproject.toml`), acessado em tempo de execução por meio de `context.run_config`. Esse parâmetro permite controlar, sem necessidade de alterar o código-fonte da função `train()`, qual fração dos clientes participantes será configurada como maliciosa em cada experimento.

A Figura 1.5 apresenta a evolução da acurácia e da perda do modelo global ao longo das rodadas de treinamento federado utilizando o protocolo de agregação FedAvg.

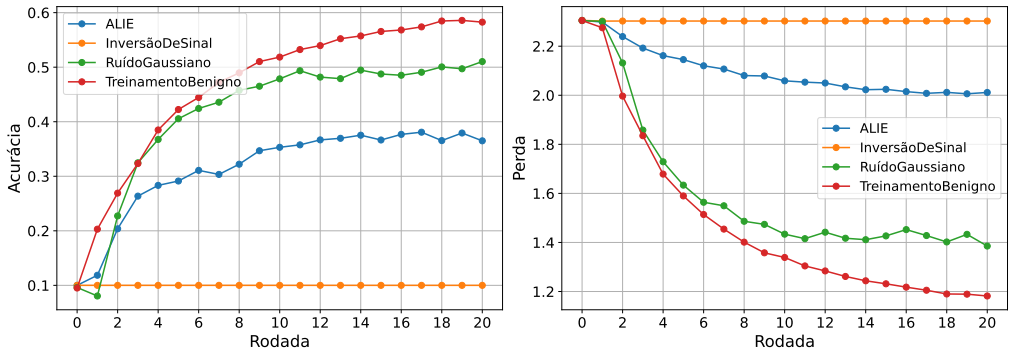


Figura 1.6. Evolução da acurácia e da perda ao longo das rodadas de treinamento sob o protocolo de agregação TrimmedMean, comparando o treinamento benigno com os ataques. Observa-se que a inversão de sinal compromete severamente a convergência, mantendo a acurácia próxima ao nível inicial e a perda praticamente constante, enquanto ALIE também degrada o treinamento de forma significativa. Em contraste, o ruído gaussiano produz impacto intermediário, e o treinamento benigno apresenta o melhor comportamento global.

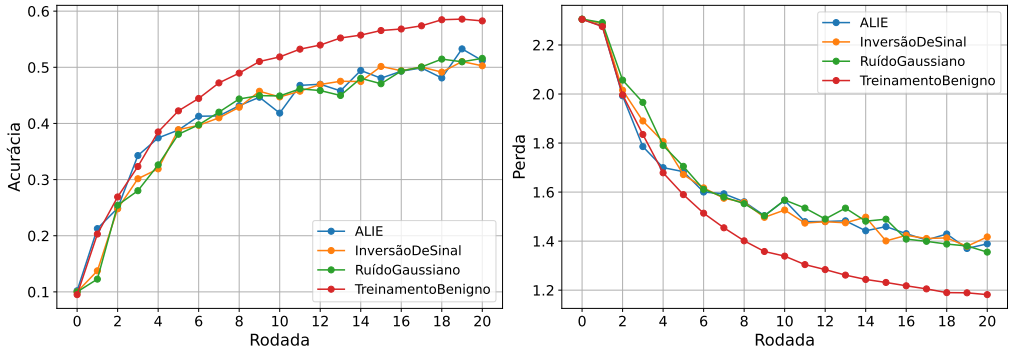


Figura 1.7. Evolução da acurácia e da perda ao longo das rodadas de treinamento sob o protocolo de agregação MultiKrum, comparando o treinamento benigno com os ataques. Diferentemente do observado com TrimmedMean, o MultiKrum reduz substancialmente o impacto dos ataques, permitindo que os cenários maliciosos mantenham trajetórias de convergência mais próximas entre si e mais próximas do treinamento benigno, ainda que com desempenho final inferior.

Observa-se que, no cenário benigno, o modelo apresenta crescimento progressivo da acurácia e redução consistente da perda, indicando convergência estável do treinamento. Em contraste, os ataques comprometem esse comportamento em diferentes níveis. O ataque por inversão de sinal é o mais severo, mantendo a acurácia próxima a 0,1 e a perda praticamente estável em valores elevados, o que indica forte comprometimento da aprendizagem global. O ataque por ruído gaussiano também degrada significativamente o desempenho, produzindo oscilações mais acentuadas e limitando o avanço da acurácia. Já o ataque ALIE, embora menos destrutivo que os demais, ainda reduz o desempenho em relação ao treinamento benigno, com acurácia inferior e perda superior ao longo das rodadas.

Na Figura 1.6, observa-se que os ataques afetam o treinamento de maneira bastante distinta. O ataque por inversão de sinal é o mais severo, pois mantém a acurácia praticamente estagnada em torno de 0,1 ao longo de todas as rodadas, ao mesmo tempo em que a perda permanece elevada e quase constante, indicando falha de convergência. O ataque ALIE também compromete significativamente o aprendizado, reduzindo a taxa de crescimento da acurácia e mantendo a perda em níveis substancialmente superiores aos do treinamento benigno. Já o ruído gaussiano produz um efeito menos destrutivo, embora

degrade o desempenho em relação ao caso benigno, ainda permite evolução do treinamento, com acurácia final superior à obtida com ALIE e perda final inferior à observada nos cenários mais agressivos.

Na Figura 1.7, o efeito dos ataques é significativamente atenuado em comparação ao caso com TrimmedMean. Todos os cenários maliciosos apresentam crescimento progressivo de acurácia e redução consistente da perda, indicando que o treinamento continua convergente mesmo sob comportamento adversarial. O treinamento benigno ainda apresenta o melhor desempenho final, com maior acurácia e menor perda, mas a distância em relação aos cenários com participantes adversariais torna-se bem menor comparada aos demais cenários.

1.5.4. Implementação dos Ataques à Privacidade

Esta seção apresenta a implementação prática dos ataques à privacidade considerados, com o objetivo de demonstrar, em ambiente controlado, como informações sensíveis podem ser inferidas a partir dos elementos compartilhados durante o treinamento federado. No ataque DLG, considera-se um servidor agregador *honesto-mas-curioso*, enquanto, na reconstrução com GAN, considera-se um cliente participante curioso.

O ataque DLG tem como objetivo reconstruir uma amostra privada de treinamento a partir dos gradientes compartilhados durante o Aprendizado Federado. O código fonte encontra-se disponível no GitLab[8]. Para executar o comportamento do servidor honesto-mas-curioso no Flower, foi criada uma nova *Strategy*, denominada `MaliciousFedAvg`, que preserva a lógica do *FedAvg*, mas acrescenta os passos do ataque antes da agregação final. A principal modificação foi feita no método `aggregate_train()`.Esse método continua responsável por agregar as respostas dos clientes, mas passa a incluir também a seleção de uma vítima, a recuperação de seus gradientes e a chamada da rotina de reconstrução DLG.

Ainda na função `aggregate_train()`, o servidor agregador verifica respostas recebidas e compara o estado do modelo global antes e depois do treinamento local. O servidor escolhe aleatoriamente um cliente como vítima, e armazena a resposta. Com isso, a implementação simula um servidor que observa normalmente todas as respostas, mas decide explorar apenas uma delas em cada rodada.

O servidor, calcula

$$g = \theta_{\text{server}} - \theta_{\text{victim}},$$

isto é, a diferença entre os parâmetros do modelo global enviados no início da rodada e os parâmetros devolvidos pelo cliente após o treinamento local. Essa etapa materializa a hipótese central do ataque: se o servidor conhece o modelo enviado e o modelo recebido, então o servidor também consegue reconstruir a atualização produzida pelo cliente, e essa atualização pode ser explorada para inferir informação privada.

O servidor chama a função `dlg_attack_from_gradients()`, função que implementa o ataque DLG, fornecendo a rede, os gradientes recuperados, a forma esperada da entrada, o número de classes, o número de iterações do *Limited-memory Broyden-Fletcher-Goldfarb-Shannon* (L-BFGS), o diretório de saída e o identificador

[8]Disponível em `https://gitlab.com/LCC/fl-dlg-attack`. Acessado em 21/04/2026.

da vítima. O L-BFGS é um método de otimização empregado para ajustar iterativamente a entrada fictícia do ataque DLG, de modo que os gradientes produzidos por essa entrada se aproximem dos gradientes reais observados [Liu e Nocedal, 1989]. Finalmente,a implementação retorna `super().aggregate_train(server_round, iter(replies_list))`, o que significa que, após executar o ataque, o servidor prossegue com a agregação padrão do *FedAvg*.

A rotina de reconstrução propriamente dita é implementada na função `dlg_attack_from_gradients()`, apresentada no Código 1.5. Essa função recebe como entrada o modelo já carregado com os parâmetros globais e a lista de gradientes recuperados da vítima, e tenta reconstruir uma entrada fictícia cuja passagem pela rede produza gradientes o mais próximos possível dos gradientes observados.

```
def dlg_attack_from_gradients(
    net, original_dy_dx,
    input_shape=(1, 3, 32, 32), num_classes=100,
    num_iterations=300, save_dir="dlg_results", client_id=None,
):
    tag = f"client_{client_id}" if client_id is not None else "victim"
    save_dir = os.path.join(save_dir, tag)
    os.makedirs(save_dir, exist_ok=True)

    device = next(net.parameters()).device
    criterion = cross_entropy_for_onehot

    print(f"\n{'='*60}")
    print(f"  Ataque DLG em {tag} - {num_iterations} iteracoes L-BFGS")
    print(f"{'='*60}")

    dummy_data = torch.randn(input_shape).to(device).requires_grad_(
    True)
    dummy_label = torch.randn((1, num_classes)).to(device).
    requires_grad_(True)
    initial_noise = tt(dummy_data[0].cpu().detach())

    optimizer = torch.optim.LBFGS([dummy_data, dummy_label])
    history, losses = [], []

    for iters in range(num_iterations):
        def closure():
            optimizer.zero_grad()
            pred = net(dummy_data)
            dummy_onehot_label = F.softmax(dummy_label, dim=-1)
            dummy_loss = criterion(pred, dummy_onehot_label)
            dummy_dy_dx = torch.autograd.grad(
                dummy_loss, net.parameters(), create_graph=True)
            grad_diff = 0
            for gx, gy in zip(dummy_dy_dx, original_dy_dx):
                grad_diff += ((gx - gy) ** 2).sum()
            grad_diff.backward()
            return grad_diff

        optimizer.step(closure)
        current_loss = closure()
```

```
40          losses.append(current_loss.item())
41          if iters % 10 == 0:
42              print(f"  {tag} | iter {iters:>3d} | loss {current_loss.
    item():.4f}")
43          history.append(tt(dummy_data[0].cpu()))
44
45      recovered_label = torch.argmax(dummy_label, dim=-1).item()
46      print(f"  {tag} | Label recuperada: {recovered_label}")
47
48      _save_results(history, losses, initial_noise, None, save_dir,
49                    f"Reconstrucao DLG - {tag}")
50      print(f"  Resultados salvos em: {os.path.abspath(save_dir)}/")
51      return dummy_data.detach(), dummy_label.detach()
```

Código Fonte 1.5. Função `dlg_attack_from_gradients()`

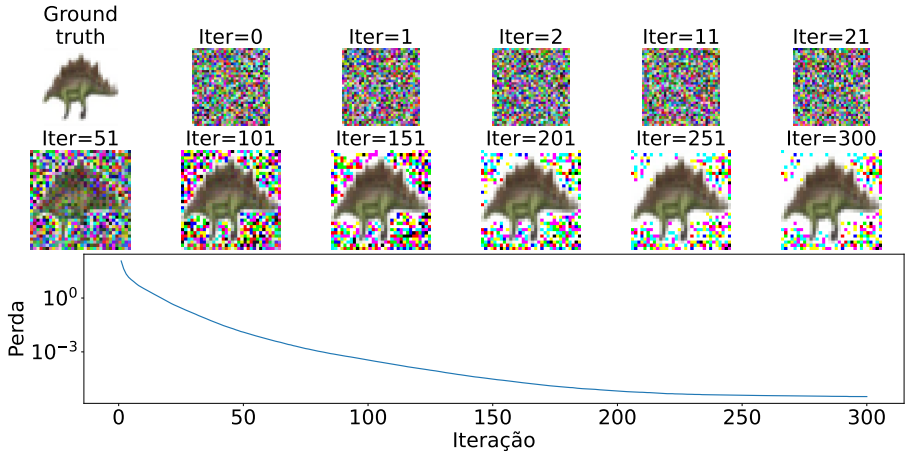
A função `dlg_attack_from_gradients()`, apresentada no Código 1.5, recebe como entrada o modelo e os gradientes recuperados e resolve o problema inverso de reconstrução. O primeiro ponto central da implementação está nas linhas 17 e 18, em que o ataque inicializa uma imagem fictícia e um rótulo fictício, ambos aleatórios, e os transforma em variáveis otimizáveis. O objetivo do DLG não é ajustar os parâmetros da rede, mas ajustar essa entrada fictícia até que ela produza gradientes compatíveis com os gradientes observados.

Na linha 21, a implementação define o L-BFGS como método de otimização. Esse otimizador é usado para atualizar simultaneamente a imagem e o rótulo fictícios ao longo das iterações do ataque. Na linha 27, a imagem fictícia atual é passada pela rede. Na linha 28, o rótulo fictício é convertido em uma distribuição válida. Na linha 29, calcula-se a perda associada a essa entrada fictícia. O ponto mais importante aparece nas linhas 30 a 32, em que o ataque calcula os gradientes fictícios da imagem atualmente reconstruída em relação aos parâmetros da rede. O argumento `create_graph=True` é indispensável, pois permite diferenciar posteriormente a discrepância entre esses gradientes fictícios e os gradientes reais em relação à própria imagem fictícia.

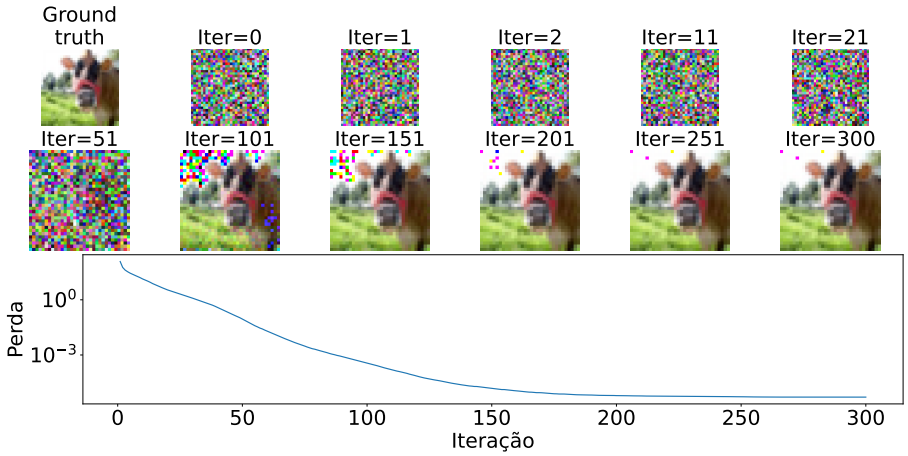
Em seguida, nas linhas 33 a 35, o código define a função objetivo do ataque, acumulando a soma das diferenças quadráticas entre os gradientes fictícios e os gradientes reais observados. Portanto, o DLG pode ser entendido como um problema de otimização em que se busca minimizar a discrepância entre dois conjuntos de gradientes. Na linha 36, a discrepância calculada é retropropagada até `dummy_data` e `dummy_label`. Esse é o mecanismo que ajusta a reconstrução, a cada iteração, a imagem fictícia é modificada na direção que reduz a diferença entre os gradientes que ela produz e os reais da vítima.

Fora da `closure()`, a linha 39 faz com que o L-BFGS atualize a imagem e o rótulo fictícios com base nessa função objetivo. Ao repetir esse processo por várias iterações, a entrada inicialmente aleatória tende a evoluir para uma amostra capaz de reproduzir os gradientes observados, o que constitui a reconstrução do ataque. Por fim, na linha 46, a implementação recupera o rótulo inferido a partir do vetor otimizado. Assim, ao final do processo, o ataque produz não apenas uma imagem reconstruída, mas também uma estimativa do rótulo correspondente.

A Figura 1.8 apresenta dois exemplos de execução do ataque DLG, um para uma amostra da classe “dinossauro” e outro para uma amostra da classe “gado”. Em ambos os



(a) Reconstrução de uma amostra da classe “dinossauro”.



(b) Reconstrução de uma amostra da classe “gado”.

Figura 1.8. Evolução do ataque DLG ao longo das iterações. O gráfico superior mostra a redução da função de perda de correspondência de gradientes, enquanto as imagens inferiores ilustram a progressão da reconstrução, partindo de ruído aleatório até uma amostra visualmente próxima da imagem original.

casos, observa-se a redução progressiva da função de perda associada à correspondência entre gradientes reais e gradientes fictícios ao longo das iterações, enquanto é mostrada a evolução visual da reconstrução, iniciando-se a partir de ruído aleatório e convergindo gradualmente para uma imagem com estrutura semântica próxima da amostra original.

Diferentemente do DLG, em que o adversário é o servidor agregador, na reconstrução com GAN o atacante é modelado como um cliente participante curioso. O Código fonte sompleto está disponível no GitLab[9]. Essa implementação se baseou exatamente na descrição do trabalho [Zhu et al., 2019].

A lógica principal do atacante é implementada na classe `AdversaryClient`, implementada em `client_app.py`[10]. Essa classe representa o cliente curioso res-

[9]Disponível em `https://gitlab.com/LCC/fl-gan-reconstruction-attack`. Acessado em 21/04/2026

[10]Disponível em `https://gitlab.com/LCC/fl-gan-reconstruction-attack/-/blob/main/gan_attack_flower/client_app.py`. Acessado em 21/04/2026

ponsável por treinar o gerador, sintetizar amostras, inseri-las em seu treinamento local e devolver ao servidor uma atualização contaminada por esse processo.

O comportamento malicioso começa no método `fit()`. Nesse método o cliente malicioso usa o modelo global recebido como o discriminador, posteriormente usado no trinamento do gerador. Em outras palavras, o modelo global funciona como a fonte de informação que incorpora conhecimento extraído dos dados privados da vítima.

Ainda no método `fit()`, o cliente chama o método `train_generator(...)`. Nesse método, o modelo global atual é usado como discriminador, e o gerador é otimizado para produzir amostras que o classificador reconheça como pertencentes à `target_class`. Em termos práticos, o atacante usa o modelo global como um oráculo diferenciável que contém informação sobre a classe privada da vítima. Quanto melhor o gerador aprender a explorar esse oráculo, mais próximas da classe-alvo tenderão a ser as imagens sintetizadas.

Posteriormente, o cliente gera um conjunto de amostras sintéticas por meio de `generate_samples(...)` e cria um vetor de rótulos artificiais usando `FAKE_CLASS_INDEX`. Esse é um dos pontos mais característicos do ataque; as amostras produzidas pela GAN tentam se parecer com a classe privada da vítima, mas são propositalmente rotuladas com uma classe artificial. Esse mecanismo implementa o engano sobre o modelo global.

O cliente, então, começa a executar o treinamento local chamando o método `train_classifier(...)` com os tensores `injected_x` e `injected_y`, gerador por meio do método `generate_samples(...)`. Essa etapa é onde as amostras geradas pela GAN deixam de ser apenas um artefato de inspeção visual e passam a interferir diretamente na atualização local devolvida ao servidor. Em outras palavras, o atacante injeta amostras sintéticas de aparência próxima à classe-alvo, mas com uma classe artificial, e força o classificador a ajustar suas fronteiras de decisão em resposta a esse estímulo. É esse ciclo de realimentação entre geração e treinamento federado que torna o ataque eficaz ao longo das rodadas.

Por fim, o cliente retorna ao servidor os parâmetros atualizados do classificador, o tamanho efetivo do conjunto de treinamento utilizado e métricas adicionais, entre elas a perda do treinamento local e a perda do gerador. O ponto importante é que o servidor recebe uma atualização aparentemente legítima, mas essa atualização já foi influenciada pela injeção adversarial de amostras sintéticas.

Na implementação, foi desenvolvido um arquivo chamado `attack.py`[11] para auxiliar no ataque. As funções principais do arquivo `attack.py` são apresentadas no Código 1.6. As funções apresentadas implementam o treinamento do gerador contra o modelo global e a geração das amostras sintéticas usadas pelo atacante.

```
1 def train_generator(
2     generator: Generator,
3     discriminator: nn.Module,
4     target_class: int,
5     latent_dim: int,
```

[11] Diponível em `https://gitlab.com/LCC/fl-gan-reconstruction-attack/-/blob/main/gan_attack_flower/attack.py`. Acessado em 21/04/2026.

```
6       steps: int,
7       batch_size: int,
8       lr: float,
9       device: torch.device,
10  ) -> float:
11      discriminator.eval()
12      for p in discriminator.parameters():
13          p.requires_grad_(False)
14
15      generator.train()
16      optimizer = torch.optim.SGD(generator.parameters(), lr=lr, momentum
        =0.0)
17      criterion = nn.NLLLoss()
18
19      target = torch.full((batch_size,), target_class, dtype=torch.long,
        device=device)
20
21      last_loss = 0.0
22      for _ in range(steps):
23          optimizer.zero_grad()
24          z = torch.randn(batch_size, latent_dim, device=device)
25          fake = generator(z)
26          log_probs = discriminator(fake)
27          loss = criterion(log_probs, target)
28          loss.backward()
29          optimizer.step()
30          last_loss = float(loss.item())
31
32      for p in discriminator.parameters():
33          p.requires_grad_(True)
34
35      return last_loss
36
37  @torch.no_grad()
38  def generate_samples(
39      generator: Generator,
40      num_samples: int,
41      latent_dim: int,
42      device: torch.device,
43  ) -> torch.Tensor:
44      generator.eval()
45      z = torch.randn(num_samples, latent_dim, device=device)
46      return generator(z)
```

Código Fonte 1.6. Funções relevantes do arquivo `attack.py`

Na função `train_generator()`, o modelo global é usado no modo avaliação para atuar apenas como fonte de sinal para o ataque, enquanto o gerador passa a ser o único componente efetivamente otimizado. Em seguida, define-se como objetivo fazer com que as amostras sintéticas produzidas pelo gerador sejam classificadas pelo modelo global como pertencentes à `target_class`, que corresponde à classe privada da vítima. O núcleo do ataque ocorre no laço iterativo (linhas 22), em que os ruídos aleatórios são convertidos em imagens sintéticas; essas imagens são submetidas ao modelo global e a perda resultante é retropropagada apenas para atualizar o gerador. Dessa forma, a cada

passo de otimização, o gerador é ajustado para produzir amostras cada vez mais compatíveis com a classe-alvo segundo o conhecimento do modelo global. Ao final, os gradientes do modelo global (linhas 32 e 33) são reabilitados para que ele volte a participar normalmente do treinamento federado nas etapas seguintes.

A função `generate_samples()` é mais simples, mas também é essencial para o ataque. Nas linhas 45 a 47, o gerador é colocado em modo de avaliação, um conjunto de ruídos aleatórios é amostrado e, na linha 47, esses vetores são convertidos em imagens sintéticas. Em conjunto, essas funções mostram que o ataque GAN em Aprendizado Federado não consiste apenas em treinar um gerador isolado. O ponto principal é a interação entre três elementos: (i) o modelo global recebido pelo cliente curioso, (ii) o treinamento do gerador para explorar a informação contida nesse modelo e (iii) a injeção das amostras geradas no treinamento local do próprio atacante. Essa combinação é que transforma o cliente curioso em um adversário capaz de reconstruir progressivamente conteúdo privado da vítima.

A Figura 1.9 compara amostras originais com as respectivas reconstruções obtidas pelo ataque. De modo geral, observa-se que as imagens reconstruídas preservam a estrutura global e os traços mais característicos da classe, embora apresentem maior desfoque e menor nitidez em relação às amostras originais. Ainda assim, a similaridade entre as duas linhas indica que o ataque foi capaz de recuperar informação visual relevante sobre o conteúdo privado, mesmo sem acesso direto aos dados reais. Esse resultado reforça que, em FL, as informações exploradas pelo atacante podem ser suficientes para reconstruir amostras com alto grau de correspondência estrutural com os dados originais.

1.6. Discussão, Tendências e Desafios de Pesquisa

O Aprendizado Federado tem sido aplicado em diferentes cenários que envolvem dados distribuídos e sensíveis, nos quais a centralização das informações é limitada por requisitos de privacidade, regulamentação ou infraestrutura. Nessa condição, sua adoção evidencia tanto o potencial do treinamento colaborativo quanto a complexidade associada à sua utilização em ambientes reais. Diferentes domínios de aplicação impõem requisitos específicos ao uso do Aprendizado Federado, influenciando diretamente os desafios enfrentados na sua implementação e as soluções investigadas na literatura. Inicialmente, apresentam-se aplicações críticas e requisitos de segurança em Aprendizado Federado, em seguida são discutidos os principais desafios em aberto e, por fim, são descritos projetos e linhas de pesquisa voltados à mitigação dessas limitações.

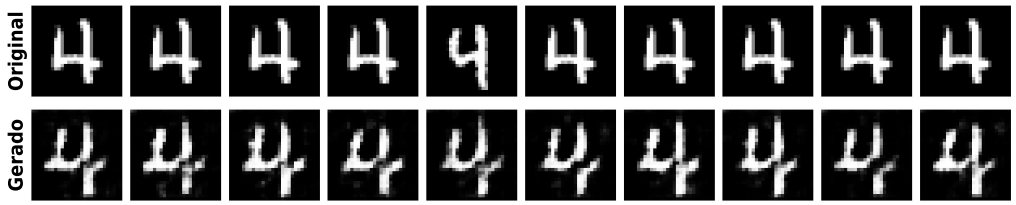


Figura 1.9. Comparação entre amostras originais e reconstruções obtidas pelo ataque. A linha superior apresenta exemplos de imagens originais, enquanto a linha inferior mostra as respectivas imagens reconstruídas, permitindo avaliar visualmente o grau de preservação da estrutura da classe alvo.

1.6.1. Aplicações críticas e requisitos de segurança em Aprendizado Federado

O Aprendizado Federado tem sido empregado em aplicações nas quais a proteção dos dados dos usuários e a conformidade com regulamentações vigentes são requisitos centrais [Wen et al., 2023]. Nesse contexto, diferentes domínios de aplicação impõem requisitos específicos para a operação segura e eficiente desse paradigma, condicionando as técnicas a serem adotadas. Na área da saúde, o Aprendizado Federado permite treinar modelos a partir de dados distribuídos entre instituições, ampliando o espaço amostral e favorecendo a generalização em tarefas como a classificação de imagens médicas [Rieke et al., 2020]. Esse domínio é marcado pela alta sensibilidade dos dados e por regulamentações rigorosas, de modo que aplicações de Aprendizado Federado em saúde demandam mecanismos de governança capazes de definir responsabilidades, regras de acesso, procedimentos de compartilhamento e critérios de uso dos modelos resultantes [Eden et al., 2025]. Nessa configuração, garantir confidencialidade, integridade das atualizações e confiabilidade do processo de treinamento torna-se essencial para evitar vazamentos de informação ou manipulações que comprometam os resultados obtidos [Nguyen et al., 2022].

No setor financeiro, aplicações como detecção de fraudes e análise de risco de crédito se beneficiam da colaboração entre instituições sem compartilhamento direto de dados sensíveis. Trata-se de um domínio altamente regulado, sujeito a normas internas e legislações de proteção de dados que restringem o trânsito de informações pessoais e transacionais [Kennedy et al., 2025]. Além da privacidade, robustez e confiabilidade dos modelos são requisitos fundamentais, pois decisões derivadas desses sistemas impactam diretamente indivíduos e organizações, exigindo garantias de integridade das contribuições e resistência a comportamentos adversariais.

Em cenários de cidades inteligentes e IoT, o Aprendizado Federado permite explorar dados provenientes de sensores e dispositivos distribuídos sem centralizar todas as medições em um único ponto. Esses ambientes são caracterizados por alta heterogeneidade, tanto em termos de distribuição dos dados quanto de capacidade computacional e de comunicação dos dispositivos participantes [Pandya et al., 2023]. Nessas condições, mecanismos flexíveis e adaptativos tornam-se necessários para lidar com limitações de processamento, variações de conectividade e diferentes níveis de desempenho, ao mesmo tempo em que requisitos de escalabilidade, eficiência de comunicação e tolerância à variabilidade dos dispositivos são determinantes para viabilizar o uso em larga escala.

De forma geral, diferentes domínios de aplicação atribuem pesos distintos a propriedades como segurança, privacidade, robustez, escalabilidade e eficiência operacional, influenciando diretamente o desenho dos sistemas de Aprendizado Federado [Pandya et al., 2023]. Essa diversidade de requisitos evidencia a complexidade de aplicar o paradigma em ambientes reais e prepara o terreno para os desafios em aberto discutidos na subseção seguinte. Os requisitos das aplicações críticas em diferentes domínios de aplicação do Aprendizado Federado são exemplificados na Tabela 1.2.

1.6.2. Desafios em Aberto

O paradigma de Aprendizado Federado pode mitigar a exposição direta de dados locais, oferecer maior flexibilidade para atender a requisitos regulatórios de proteção de

Tabela 1.2. Exemplos de domínios de aplicação de Aprendizado Federado e requisitos associados.

Domínio	Principais requisitos	Impacto dos requisitos
Saúde	Privacidade forte, conformidade regulatória, governança, integridade das atualizações, robustez a ataques	Restringe o acesso a dados brutos, exige mecanismos de proteção contra vazamento e manipulação, e condiciona o uso clínico dos modelos a critérios de confiança e auditabilidade.
Setor financeiro	Privacidade de dados sensíveis, conformidade com normas e leis, robustez, confiabilidade decisória	Impõe alta exigência de segurança e resiliência a ataques, pois erros ou manipulações podem afetar diretamente decisões de crédito, detecção de fraudes e riscos institucionais.
Cidades inteligentes e IoT	Escalabilidade, eficiência de comunicação, tolerância a heterogeneidade, privacidade, robustez	Exige protocolos leves e adaptativos que operem com dispositivos limitados e redes instáveis, mantendo proteção de dados e desempenho aceitável em grande escala.

dados e viabilizar a colaboração entre diferentes instituições no treinamento de modelos. Apesar dessas vantagens, a adoção prática ainda enfrenta desafios importantes, pois a necessidade de garantir segurança, privacidade, confiabilidade e integridade de dados e modelos impõe restrições significativas ao uso em ambientes reais [Zhao et al., 2025]. Esses obstáculos extrapolam o nível algorítmico e envolvem também limitações de infraestrutura, heterogeneidade entre participantes e restrições operacionais típicas de sistemas distribuídos [Kairouz e McMahan, 2021].

Um desafio central em Aprendizado Federado é a heterogeneidade inerente a ambientes descentralizados. Essa heterogeneidade manifesta-se, principalmente, na forma de heterogeneidade estatística dos dados, frequentemente associada ao cenário não independente e não identicamente distribuído (*non-IID*), e na heterogeneidade de recursos e condições dos próprios clientes. A heterogeneidade estatística [Lu et al., 2024] pode comprometer a convergência e a capacidade de generalização do modelo global, enquanto as diferenças em capacidade computacional, disponibilidade de recursos e qualidade de conexão de rede introduzem desequilíbrios no treinamento, como atrasos de atualização e ociosidade decorrente de clientes lentos [Ye et al., 2023]. Embora essas variações possam enriquecer a diversidade dos dados, a combinação de distribuição *non-IID* e heterogeneidade de sistema tende a dificultar a convergência, aumentar o custo de comunicação e reduzir a qualidade final do modelo em cenários práticos [Lu et al., 2024, Ye et al., 2023].

Outro desafio amplamente discutido refere-se ao equilíbrio entre privacidade, custo e desempenho em Aprendizado Federado. A redução da exposição direta dos dados dos clientes é uma das principais motivações para adoção do paradigma federado, o que torna essencial a proteção contra vazamentos ou inferência de informações sensíveis [Zhao et al., 2025]. Nesse contexto, técnicas como privacidade diferencial e *secure aggregation* são empregadas para mitigar riscos de exposição dos dados locais [Geyer et al., 2017, Bonawitz et al., 2017], mas introduzem um compromisso entre privacidade, utilidade e custos computacionais. A privacidade diferencial adiciona ruído aos dados ou às atualizações, podendo deteriorar a acurácia e a eficiência do modelo global [Geyer

et al., 2017], enquanto a *secure aggregation* eleva o custo computacional e de comunicação e restringe o acesso a contribuições individuais dos clientes [Bonawitz et al., 2017]. Como consequência, essas camadas de proteção reduzem a visibilidade sobre o processo de treinamento, dificultam a análise de falhas, vieses e comportamentos maliciosos e podem levar a modelos globais com desempenho degradado, vieses indesejados ou comportamentos inconsistentes [Zhao et al., 2025].

A literatura também identifica como desafios atuais e relevantes ataques de envenenamento (*poisoning*), ataques Bizantinos e ataques *backdoor*, nos quais clientes maliciosos enviam atualizações manipuladas com o objetivo de degradar o modelo global ou induzir comportamentos específicos [Zhao et al., 2025,Chen et al., 2022]. Como resposta, técnicas de agregação robusta têm sido propostas para reduzir a influência de atualizações inconsistentes ou maliciosas no treinamento federado [Pillutla et al., 2022]. Ainda assim, essas abordagens apresentam limitações importantes em cenários com dados *non-IID*, nos quais a variabilidade legítima entre clientes pode mimetizar comportamentos adversariais, tornando mais difícil distinguir contribuições benignas de ações maliciosas [Lu et al., 2024, Pillutla et al., 2022].

Além desses aspectos, o Aprendizado Federado enfrenta desafios de escalabilidade e restrições operacionais dos sistemas distribuídos. A diversidade de características dos clientes, como capacidade de processamento, qualidade de conexão de rede e limitações de memória, impacta diretamente a eficiência do treinamento colaborativo e pode restringir o uso de Aprendizado Federado em cenários com requisitos rígidos de qualidade de serviço (QoS) [Ye et al., 2023]. Em ambientes reais, a heterogeneidade de desempenho entre clientes dificulta a sincronização das atualizações e a participação equilibrada, resultando em atrasos na agregação, aumento do tempo total de treinamento e maior ociosidade do sistema devido a participantes mais lentos, além de problemas de conectividade que levam à perda ou reenvio de atualizações [Ye et al., 2023]. Nessa perspectiva, o desenvolvimento de técnicas que ofereçam maior escalabilidade e flexibilidade nos requisitos impostos aos clientes permanece como um desafio crucial para a adoção do Aprendizado Federado em larga escala.

Os desafios de Aprendizado Federado precisam ser avaliados de forma integrada.Em particular, abordagens que fortalecem privacidade, robustez ou eficiência podem, simultaneamente, afetar negativamente o desempenho, a visibilidade sobre o processo de treinamento ou a escalabilidade, evidenciando a necessidade de uma análise conjunta entre objetivos e restrições [Kairouz e McMahan, 2021]. Embora muitas propostas da literatura ainda tratem problemas de forma isolada, observa-se um movimento crescente de modularização das soluções, permitindo sua adaptação e composição em diferentes arquiteturas e cenários de aplicação; essa tendência aponta para avanços não apenas na mitigação de reveses específicos, mas também na capacidade de integrar mecanismos especializados de maneira flexível e ajustada a cada cenário [Kairouz e McMahan, 2021]. A Tabela 1.3 elenca os principais desafios e propostas de mitigação.

1.6.3. Projetos de Pesquisa

Diante dos desafios de FL, diversas propostas na literatura têm buscado mitigar as suas limitações por meio de abordagens voltadas à privacidade, robustez e eficiência

Tabela 1.3. Principais desafios em Aprendizado Federado, dimensões afetadas e formas de mitigação.

Desafio	Dimensão	Mitigação
Heterogeneidade *non-IID* e impacto na convergência	Dados	Ajustes de protocolo e algoritmos de otimização/agregação adaptados a cenários *non-IID* para reduzir desequilíbrios entre contribuições [Lu et al., 2024, Pillutla et al., 2022].
Heterogeneidade de recursos dos clientes	Clientes	Estratégias assíncronas ou parciais e requisitos mais flexíveis de participação para acomodar diferenças de capacidade e conectividade [Ye et al., 2023].
Equilíbrio entre privacidade, custo e desempenho	Dados/ Agregação	Uso de privacidade diferencial e *secure aggregation* com calibração de ruído e de protocolo para limitar vazamento mantendo utilidade aceitável [Geyer et al., 2017, Bonawitz et al., 2017, Lyu, 2020].
Ataques de envenenamento, Bizantinos e *backdoor*	Dados/ Agregação	Agregação robusta, filtragem de clientes suspeitos e técnicas específicas de detecção e remoção de *backdoors* [Pillutla et al., 2022, Chen et al., 2022, Li et al., 2022b].
Escalabilidade e restrições de sistemas distribuídos	Servidor/ Seleção	Controle do número de clientes por rodada, técnicas de comunicação eficientes e algoritmos tolerantes a falhas e atrasos [Ye et al., 2023, Kairouz e McMahan, 2021].
Interdependência entre privacidade, robustez, desempenho e escalabilidade	Sistema	Desenho modular de defesas e análise conjunta de *trade-offs* para combinar mecanismos especializados de forma flexível [Kairouz e McMahan, 2021].

dos sistemas [Kairouz e McMahan, 2021]. Essas iniciativas evidenciam a natureza multiobjetivo do problema em ambientes distribuídos, indicando que soluções eficazes devem considerar as particularidades de cada cenário de aplicação e seus respectivos requisitos.

Um avanço recente nesse contexto é o desenvolvimento de metodologias de avaliação mais estruturadas, voltadas à análise sistemática de ataques e mecanismos de defesa [Han et al., 2024]. A criação de *benchmarks* padronizados tem aumentado a comparabilidade entre diferentes abordagens, permitindo avaliações mais consistentes dos métodos propostos e favorecendo a reprodutibilidade dos experimentos, o que facilita a validação de resultados e a comparação entre soluções baseadas em estratégias distintas. Paralelamente, o desenvolvimento de ferramentas e plataformas específicas tem desempenhado papel central na viabilização prática do modelo federado.

Entre essas iniciativas, destaca-se o projeto OpenMined, responsável pela biblioteca PySyft, destinada à implementação de Aprendizado Federado com ênfase em privacidade [Nguyen et al., 2024]. A PySyft permite treinar modelos sem acesso direto aos dados locais, incorporando mecanismos que viabilizam o compartilhamento seguro de informações entre entidades participantes. Outra iniciativa relevante é o Flower, um *framework* projetado para ser simples e flexível, permitindo a implementação de sistemas federados de forma agnóstica à biblioteca de aprendizado de máquina subjacente e possibilitando a orquestração de ambientes federados simulados e aplicações reais [Nguyen et al., 2024].

Além dessas bibliotecas, o Aprendizado Federado já está presente em aplicações comerciais consolidadas. Um exemplo notório é o teclado virtual do Google (Gboard), que utiliza Aprendizado Federado para aprimorar a predição de texto diretamente nos dispositivos dos usuários, treinando modelos locais a partir dos padrões de escrita individuais e compartilhando apenas atualizações de modelo com o servidor, o que contribui para pre-

servar a privacidade das informações sensíveis [Hard et al., 2018]. Esse tipo de aplicação ilustra a viabilidade do paradigma federado em larga escala quando há alinhamento entre requisitos de privacidade e capacidade de infraestrutura.

De forma geral, observa-se que muitas soluções propostas ainda se concentram em desafios específicos e são desenvolvidas como mecanismos especializados voltados a problemas particulares. Nesse cenário, ganham destaque abordagens modularizadas, que promovem maior flexibilidade de adoção e integração em diferentes arquiteturas e fluxos de treinamento [Kairouz e McMahan, 2021]. Esse movimento aponta para uma linha de pesquisa orientada à composição de mecanismos especializados, buscando equilibrar, de maneira adaptável, múltiplos requisitos do sistema em cenários reais de FL.

1.7. Considerações Finais

O Aprendizado Federado (*Federated Learning* – FL) é um paradigma promissor para treinamento colaborativo de modelos em ambientes com dados distribuídos e sensíveis, oferecendo uma alternativa à centralização tradicional ao reduzir a exposição direta de informações e facilitar a conformidade com requisitos regulatórios. Ao longo deste capítulo, foi discutido como esse paradigma também conta com uma superfície de ataque mais ampla, em que vulnerabilidades ligadas a envenenamento de modelo, ataques Bizantinos, *backdoors* e diferentes formas de quebra de privacidade colocam em risco tanto o desempenho quanto a confidencialidade dos modelos treinados. A análise dessas ameaças evidencia que a segurança em Aprendizado Federado não é um problema pontual, mas um componente intrínseco ao projeto do sistema.

No eixo da privacidade, foram discutidos ataques de inferência de participação, inferência de atributos, inferência de origem e reconstrução de conteúdo, todos explorando o fato de que atualizações de modelo, gradientes e representações internas carregam sinais sobre os dados locais. Estratégias como privacidade diferencial, criptografia homomórfica, agregação segura e computação multipartidária têm mostrado potencial para limitar esse vazamento, ainda que introduzam compromissos importantes entre proteção de privacidade, custo computacional, custo de comunicação e acurácia do modelo. Em cenários práticos, o desafio deixa de ser apenas "proteger ao máximo" e passa a envolver o ajuste fino de parâmetros e mecanismos para atingir níveis de proteção compatíveis com as exigências regulatórias e os requisitos da aplicação.

No eixo da robustez, foram caracterizados ataques de envenenamento de dados, envenenamento de modelo, ataques Bizantinos e ataques direcionados (*backdoor*), incluindo variantes por substituição de modelo, casos raros e formulações baseadas em otimização ou aprendizado por reforço. Técnicas de agregação robusta, como Krum, *Trimmed Mean*, Bulyan e RFA, bem como métodos de filtragem de clientes suspeitos foram apresentadas como defesas centrais contra ataques não-direcionados. Para ataques direcionados, defesas especializadas de detecção e eliminação, baseadas em assinaturas espectrais, agrupamento de ativações, *fine-pruning*, distilação de atenção, reparo por conectividade de modos e aprendizagem anti-*backdoor*, exemplificam a necessidade de mecanismos que operem sobre a estrutura interna do modelo e não apenas sobre estatísticas agregadas.

Em termos de desafios sistêmicos, foram destacados os efeitos da heterogeneidade estatística de dados (dados *non-IID*) e da heterogeneidade de recursos entre clientes, que

impactam a convergência, a estabilidade e o custo de treinamento. Esses fatores interagem de forma não trivial com os mecanismos de segurança, já que agregadores robustos podem confundir variabilidade legítima com comportamento malicioso, mecanismos de proteção de privacidade reduzem a visibilidade sobre contribuições individuais, e restrições de dispositivos e comunicação limitam a complexidade das defesas viáveis em produção. Como consequência, o projeto de sistemas de Aprendizado Federado em ambientes reais exige considerar simultaneamente dimensões de dados, clientes, seleção de participantes, agregação e papel do servidor, com atenção às relações de compromisso entre privacidade, robustez, desempenho e escalabilidade.

As aplicações críticas discutidas, como saúde, finanças, cidades inteligentes, IoT e serviços móveis, reforçam que não existe uma configuração única de segurança adequada a todos os cenários. Em alguns domínios, propriedades como confidencialidade, governança e auditabilidade são prioritárias, enquanto em outros a viabilidade do Aprendizado Federado depende principalmente de requisitos de Qualidade de Serviço (*Quality of Service* – QoS), eficiência e tolerância à heterogeneidade. O levantamento de projetos e *frameworks*, como PySyft e Flower, e o exemplo de uso em larga escala no Gboard da Google mostram que já existem ferramentas práticas para integrar mecanismos de segurança ao ciclo de desenvolvimento, experimentação e implantação de soluções federadas, aproximando a pesquisa acadêmica de contextos operacionais.

Observa-se que a maior parte das contribuições atuais ainda aborda problemas específicos de forma isolada, mas há um movimento crescente em direção a arquiteturas modulares que permitam combinar, de forma configurável, mecanismos de privacidade, agregação robusta, monitoramento e detecção de anomalias. Essa tendência indica que os próximos avanços relevantes em segurança para Aprendizado Federado deverão partir menos de técnicas únicas e mais de composições ajustadas, capazes de adaptar o nível de proteção às necessidades do domínio de aplicação, aos riscos aceitáveis e às limitações de infraestrutura presentes em ambientes reais.

Referências

[Agarwal et al., 2018] Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S. e McMahan, B. (2018). cpsgd: Communication-efficient and Differentially-private Distributed Sgd. *Advances in neural information processing systems*, 31.

[Blanchard et al., 2017] Blanchard, P., El Mhamdi, E. M., Guerraoui, R. e Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30.

[Bochie et al., 2021] Bochie, K., Sammarco, M., Detyniecki, M. e Campista, M. E. M. (2021). Análise do aprendizado federado em redes móveis. Em *SBRC*, p. 71–84. SBC.

[Bonawitz et al., 2017] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A. e Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. Em *Proceedings of the 2017 ACM SIGSAC*, CCS '17, p. 1175–1191, New York, NY, USA. ACM.

[Brendan McMahan et al., 2017] Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S. e Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. Em *Proceedings of the 20th AISTATS*, volume 54.

[Cao et al., 2020] Cao, X., Fang, M., Liu, J. e Gong, N. Z. (2020). fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*.

[Cao et al., 2021] Cao, X., Jia, J. e Gong, N. Z. (2021). Provably secure federated learning against malicious clients. Em *Proceedings of the AAAI conference on artificial intelligence*, volume 35, p. 6885–6893.

[Chen et al., 2018] Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I. e Srivastava, B. (2018). Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*.

[Chen et al., 2022] Chen, Y., Gui, Y., Lin, H., Gan, W. e Wu, Y. (2022). Federated learning attacks and defenses: a survey. Em *2022 IEEE international conference on big data (big data)*, p. 4256–4265. IEEE.

[Cunha Neto et al., 2023] Cunha Neto, H. N., Hribar, J., Dusparic, I., Mattos, D. M. F. e Fernandes, N. C. (2023). A survey on securing federated learning: Analysis of applications, attacks, challenges, and trends. *IEEE Access*, 11:41928–41953.

[de Assis et al., 2023] de Assis, F. M., Macedo, E. L. e de Moraes, L. F. (2023). Aplicaçao de criptografia homomórfica na mineraçao de dados em fluxos de roteadores de borda na internet. Em *SBSeg*, p. 273–278. SBC.

[de Assis et al., 2024] de Assis, F. M., Macedo, E. L. e de Moraes, L. F. (2024). Métodos para criptografia homomórfica na mineraçao de dados aplicados em fluxos de roteadores de borda na internet. Em *WGRS*, p. 70–83. SBC.

[Diana et al., 2025] Diana, F., Marfoq, O., Xu, C., Neglia, G., Giroire, F. e Thomas, E. (2025). Attribute inference attacks for federated regression tasks. Em *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, p. 16271–16279.

[Dwork e Roth, 2014] Dwork, C. e Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3-4):211–487.

[Eden et al., 2025] Eden, R., Chukwudi, I., Bain, C., Barbieri, S., Callaway, L., de Jersey, S., George, Y., Gorse, A.-D., Lawley, M., Marendy, P. et al. (2025). A scoping review of the governance of federated learning in healthcare. *npj Digital Medicine*, 8(1):427.

[Fang et al., 2020] Fang, M., Cao, X., Jia, J. e Gong, N. Z. (2020). Local model poisoning attacks to byzantine-robust federated learning. Em *29th USENIX*, p. 1623–1640.

[Fang et al., 2024] Fang, M., Zhang, Z., Hairi, Khanduri, P., Liu, J., Lu, S., Liu, Y. e Gong, N. (2024). Byzantine-robust decentralized federated learning. Em *Proceedings of the 2024 on ACM SIGSAC*, p. 2874–2888.

[Fu et al., 2022] Fu, C., Zhang, X., Ji, S., Chen, J., Wu, J., Guo, S., Zhou, J., Liu, A. X. e Wang, T. (2022). Label inference attacks against vertical federated learning. Em *31st USENIX security symposium (USENIX Security 22)*, p. 1397–1414.

[Fung et al., 2018] Fung, C., Yoon, C. J. e Beschastnikh, I. (2018). Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*.

[Geyer et al., 2017] Geyer, R. C., Klein, T. e Nabi, M. (2017). Differentially private federated learning: a client level perspective. *arXiv preprint arXiv:1712.07557*.

[Goldblum et al., 2022] Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Madry, A., Li, B. e Goldstein, T. (2022). Dataset security for machine learning: data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1.

[Guerraoui et al., 2018] Guerraoui, R., Rouault, S. et al. (2018). The hidden vulnerability of distributed learning in byzantium. Em *International conference on machine learning*, p. 3521–3530. PMLR.

[Han et al., 2024] Han, S., Buyukates, B., Hu, Z., Jin, H., Jin, W., Sun, L., Wang, X., Wu, W., Xie, C., Yao, Y. et al. (2024). Fedsecurity: a benchmark for attacks and defenses in federated learning and federated llms. Em *30th ACM SIGKDD*, p. 5070–5081.

[Hard et al., 2018] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C. e Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

[Hitaj et al., 2017] Hitaj, B., Ateniese, G. e Perez-Cruz, F. (2017). Deep models under the gan: Information leakage from collaborative deep learning. Em *2017 ACM SIGSAC*, CCS '17, p. 603–618, New York, NY, USA. ACM.

[Hu et al., 2023] Hu, H., Zhang, X., Salcic, Z., Sun, L., Choo, K.-K. R. e Dobbie, G. (2023). Source inference attacks: Beyond membership inference attacks in federated learning. *IEEE Transactions on Dependable and Secure Computing*, 21(4):3012–3029.

[Jayaraman e Evans, 2022] Jayaraman, B. e Evans, D. (2022). Are attribute inference attacks just imputation? Em *2022 ACM SIGSAC*, p. 1569–1582.

[Ji et al., 2025] Ji, A., Bandyopadhyay, B., Song, C., Krishnaswami, N., Vashisht, P., Smiroldo, R., Litton, I., Mahinder, S., Chitnis, M. e Hill, A. W. (2025). Private federated learning in real world application – a case study.

[Kairouz e McMahan, 2021] Kairouz, P. e McMahan, H. B. (2021). Advances and open problems in federated learning. *Foundations and trends in ML*, 14(1-2):1–210.

[Kennedy et al., 2025] Kennedy, C. H., Hilal, A. e Momeni, M. (2025). The role of federated learning in improving financial security: a survey. Em *2025 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*, p. 1–8. IEEE.

[Li et al., 2022a] Li, H., Sun, X. e Zheng, Z. (2022a). Learning to attack federated learning: a model-based reinforcement learning attack framework. *Advances in Neural Information Processing Systems*, 35:35007–35020.

[Li et al., 2023] Li, S., Ngai, E. C.-H. e Voigt, T. (2023). An experimental study of byzantine-robust aggregation schemes in federated learning. *IEEE Transactions on Big Data*, 10(6):975–988.

[Li et al., 2022b] Li, Y., Jiang, Y., Li, Z. e Xia, S.-T. (2022b). Backdoor learning: a survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22.

[Li et al., 2021a] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B. e Ma, X. (2021a). Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912.

[Li et al., 2021b] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B. e Ma, X. (2021b). Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*.

[Lim et al., 2020] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y. C., Yang, Q., Niyato, D. e Miao, C. (2020). Federated learning in mobile edge networks: a comprehensive survey. *IEEE Communications Surveys Tutorials*.

[Liu e Nocedal, 1989] Liu, D. C. e Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528.

[Liu et al., 2018] Liu, K., Dolan-Gavitt, B. e Garg, S. (2018). Fine-pruning: Defending against backdooring attacks on deep neural networks. Em *International symposium on research in attacks, intrusions, and defenses*, p. 273–294. Springer.

[Liu et al., 2022] Liu, P., Xu, X. e Wang, W. (2022). Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives. *Cybersecurity*, 5(1):1–19.

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

[Lyu, 2020] Lyu, L. (2020). Lightweight crypto-assisted distributed differential privacy for privacy-preserving distributed learning. Em *2020 IJCNN*, p. 1–8. IEEE.

[Lyu et al., 2022] Lyu, L., Yu, H., Ma, X., Chen, C., Sun, L., Zhao, J., Yang, Q. e Yu, P. S. (2022). Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*, 35(7):8726–8746.

[Lyu et al., 2020] Lyu, L., Yu, H. e Yang, Q. (2020). Threats to federated learning: a survey. *arXiv preprint arXiv:2003.02133*.

[McMahan et al., 2016] McMahan, H. B., Moore, E., Ramage, D. e y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2(2):15–18.

[McMahan et al., 2018] McMahan, H. B., Ramage, D., Talwar, K. e Zhang, L. (2018). Learning differentially private recurrent language models. Em *International Conference on Learning Representations*.

[Melis et al., 2019] Melis, L., Song, C., De Cristofaro, E. e Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. Em *2019 IEEE symposium on security and privacy (SP)*, p. 691–706. IEEE.

[Nasr et al., 2019] Nasr, M., Shokri, R. e Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. Em *2019 IEEE symposium on security and privacy (SP)*, p. 739–753. IEEE.

[Nguyen et al., 2022] Nguyen, D. C., Pham, Q.-V., Pathirana, P. N., Ding, M., Seneviratne, A., Lin, Z., Dobre, O. e Hwang, W.-J. (2022). Federated learning for smart healthcare: A survey. *ACM Comput. Surv.*, 55(3).

[Nguyen et al., 2024] Nguyen, G., Sainz-Pardo Diaz, J., Calatrava, A., Berberi, L., Lytvyn, O., Kozlov, V., Tran, V., Moltó, G. e Lopez Garcia, A. (2024). Landscape of machine learning evolution: Privacy-preserving federated learning frameworks and tools: G. nguyen et al. *Artificial Intelligence Review*, 58(2):51.

[Nowroozi et al., 2025] Nowroozi, E., Haider, I., Taheri, R. e Conti, M. (2025). Federated learning under attack: Exposing vulnerabilities through data poisoning attacks in computer networks. *IEEE Trans. on Net. and Service Management*, 22(1):822–831.

[Pandya et al., 2023] Pandya, S. et al. (2023). Federated learning for smart cities: a comprehensive survey. *Sustainable Energy Technologies and Assessments*, 55:102987.

[Pang et al., 2025] Pang, X., Zhao, C., Wang, Z., Hu, J., Wang, Y., Wang, L., Wei, T., Ren, K. e Chen, C. (2025). poisafl: scalable poisoning attack framework to byzantine-resilient semi-asynchronous federated learning. Em *USENIX Security*, p. 6461–6479.

[Pillutla et al., 2022] Pillutla, K., Kakade, S. M. e Harchaoui, Z. (2022). Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154.

[Rieke et al., 2020] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K. et al. (2020). The future of digital health with federated learning. *NPJ digital medicine*, 3(1):119.

[Sattler et al., 2020] Sattler, F., Müller, K.-R., Wiegand, T. e Samek, W. (2020). On the byzantine robustness of clustered federated learning. Em *IEEE ICASSP 2020*, p. 8861–8865. IEEE.

[Shejwalkar e Houmansadr, 2021] Shejwalkar, V. e Houmansadr, A. (2021). Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. Em *Ndss*.

[Shen et al., 2016] Shen, S., Tople, S. e Saxena, P. (2016). Auror: Defending against poisoning attacks in collaborative deep learning systems. Em *Proceedings of the 32nd ACSAC '16*, p. 508–519, New York, NY, USA. ACM.

[Shi et al., 2022] Shi, J., Wan, W., Hu, S., Lu, J. e Zhang, L. Y. (2022). Challenges and approaches for mitigating byzantine attacks in federated learning. Em *2022 IEEE International Conference TrustCom*, p. 139–146. IEEE.

[Shokri et al., 2017] Shokri, R., Stronati, M., Song, C. e Shmatikov, V. (2017). Membership inference attacks against machine learning models. Em *2017 IEEE symposium on security and privacy (SP)*, p. 3–18. IEEE.

[Sun et al., 2021] Sun, L., Qian, J. e Chen, X. (2021). Ldp-fl: Practical private aggregation in federated learning with local differential privacy. Em *Proceedings of the Thirtieth IJCAI-21*, p. 1571–1578. Inter. Joint Conf. on Artificial Intell. Organization.

[Tran et al., 2018] Tran, B., Li, J. e Madry, A. (2018). Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31.

[Truex et al., 2019] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R. e Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. Em *Proceedings of the 12th ACM workshop on artificial intelligence and security*, p. 1–11.

[Truex et al., 2020] Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E. e Wei, W. (2020). ldp-fed: Federated Learning with Local Differential Privacy. Em *Proceedings of the third ACM inter. workshop on edge systems, analytics and networking*, p. 61–66.

[Wang et al., 2020] Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K. e Papailiopoulos, D. (2020). Attack of the tails: Yes, you really can backdoor federated learning. Em *Advances in Neural Information Processing Systems*, volume 33, p. 16070–16084. Curran Associates, Inc.

[Wang et al., 2019] Wang, N., Xiao, X., Yang, Y., Zhao, J., Hui, S. C., Shin, H., Shin, J. e Yu, G. (2019). Collecting and analyzing multidimensional data with local differential privacy. Em *2019 IEEE 35th ICDE*, p. 638–649. IEEE.

[Wang et al., 2025] Wang, T.-H., Chen, P.-N. e Huang, Y.-C. (2025). Probabilistic byzantine attack on federated learning. *IEEE Transactions on Signal Processing*.

[Wang et al., 2022] Wang, Z., Huang, Y., Song, M., Wu, L., Xue, F. e Ren, K. (2022). Poisoning-assisted property inference attack against federated learning. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3328–3340.

[Wen et al., 2023] Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J. e Zhang, W. (2023). A survey on federated learning: Challenges and applications. *International journal of machine learning and cybernetics*, 14(2):513–535.

[Wu et al., 2024] Wu, D., Bai, J., Song, Y., Chen, J., Zhou, W., Xiang, Y. e Sajjanhar, A. (2024). fedinverse: evaluating privacy leakage in federated learning. Em *The twelfth international conference on learning representations*.

[Xia et al., 2024] Xia, F., Liu, Y., Jin, B., Yu, Z., Cai, X., Li, H., Zha, Z., Hou, D. e Peng, K. (2024). Leveraging multiple adversarial perturbation distances for enhanced membership inference attack in federated learning. *Symmetry*, 16(12):1677.

[Xie et al., 2020] Xie, C., Koyejo, S. e Gupta, I. (2020). Zeno++: Robust fully asynchronous sgd. Em *Inter. Conference on Machine Learning*, p. 10495–10503. PMLR.

[Yaldiz et al., 2023] Yaldiz, D. N., Zhang, T. e Avestimehr, S. (2023). Secure federated learning against model poisoning attacks via client filtering. *arXiv preprint arXiv:2304.00160*.

[Yang et al., 2025] Yang, L., Miao, Y., Liu, Z., Liu, Z., Li, X., Kuang, D., Li, H. e Deng, R. H. (2025). Enhanced model poisoning attack and multi-strategy defense in federated learning. *IEEE Transactions on Information Forensics and Security*.

[Yang et al., 2019] Yang, Q., Liu, Y., Chen, T. e Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Trans. on Intell. Sys. and Tech.*, 10(2):1–19.

[Ye et al., 2023] Ye, M. et al. (2023). Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44.

[Yin et al., 2018] Yin, D., Chen, Y., Kannan, R. e Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. Em *International conference on machine learning*, p. 5650–5659. Pmlr.

[Zhang et al., 2019] Zhang, J., Chen, J., Wu, D., Chen, B. e Yu, S. (2019). Poisoning attack in federated learning using generative adversarial nets. Em *2019 18th IEEE TrustCom/BigDataSE*, p. 374–380.

[Zhang et al., 2025] Zhang, K., Xu, P. e Tian, Z. (2025). Distributional and byzantine robust decentralized federated learning. Em *2025 59th Annual CISS*, p. 1–6. IEEE.

[Zhao et al., 2020a] Zhao, B., Mopuri, K. R. e Bilen, H. (2020a). idlg: improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*.

[Zhao et al., 2025] Zhao, J. et al. (2025). The federation strikes back: a survey of federated learning privacy attacks, defenses, applications, and policy landscape. *ACM Computing Surveys*, 57(9):1–37.

[Zhao et al., 2020b] Zhao, P., Chen, P.-Y., Das, P., Ramamurthy, K. N. e Lin, X. (2020b). Bridging mode connectivity in loss landscapes and adversarial robustness. *arXiv preprint arXiv:2005.00060*.

[Zhao et al., 2020c] Zhao, Y. et al. (2020c). Local differential privacy-based federated learning for internet of things. *IEEE IoT Journal*, 8(11):8836–8853.

[Zhu et al., 2025] Zhu, G. et al. (2025). fedmia: an effective membership inference attack exploiting "all for one" principle in federated learning. Em *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 20643–20653.

[Zhu et al., 2019] Zhu, L., Liu, Z. e Han, S. (2019). Deep leakage from gradients. *Advances in neural information processing systems*, 32.

Capítulo 2

Aprendizado Federado Veicular: da Teoria à Prática

Lucas Airam Castro de Souza (UFRJ e INRIA), Guilherme Araujo Thomaz (UFRJ), Mateus da Silva Gilbert (UFRJ), Vinicius de Oliveira Avena (UFRJ), Felipe Gomes Táparo (UFRJ), João Victor Dias Sobrinho (UFRJ), Fernando Dias de Mello Silva (UFRJ), Nadjib Achir (INRIA), Miguel Elias Mitre Campista (UFRJ), Luís Henrique Maciel Kosmalski Costa (UFRJ)

Resumo

Veículos conectados e automatizados (Connected and Autonomous Vehicles - CAVs) aprimoram a experiência de condução e o conforto dos passageiros por meio de diversas aplicações de segurança e entretenimento, muitas delas baseadas em aprendizado de máquina. Diante da necessidade de utilizar dados pessoais para o aperfeiçoamento dessas tecnologias, o aprendizado federado surge como uma alternativa para treinar modelos de aprendizado de máquina com garantia de privacidade. Entretanto, devido ao ambiente dinâmico com alta mobilidade, o aprendizado federado veicular (Vehicular Federated Learning – VFL) apresenta desafios além dos inerentes ao FL tradicional. Este minicurso contextualiza a execução básica e a relevância do VFL, introduzindo sua arquitetura, detalhando estratégias para superar os desafios de latência e mobilidade enfrentados no cenário veicular. Além disso, o minicurso discute a infraestrutura de rede necessária para o VFL, analisando o plano de dados e o plano de controle na transmissão de modelos e a orquestração de tarefas. O conteúdo também abrange as tendências propostas para a superação dos desafios práticos, incluindo a heterogeneidade de dispositivos dos clientes, as limitações de largura de banda do canal sem fio e a distribuição estatística de dados entre os veículos. Ainda, o presente minicurso pretende aprofundar-se e elevar a experiência dos participantes de forma prática, demonstrando como implementar e testar soluções. Por fim, o minicurso discute pontos em aberto e futuras direções de pesquisa.

2.1. Introdução

Veículos conectados e automatizados (*Connected and Autonomous Vehicles* - CAVs) transformam a mobilidade por meio da integração de sensores e atuadores, com-

binando com a inteligência artificial. Os veículos autônomos são divididos em 6 níveis de acordo com as suas capacidades, conforme exibido na Figura 2.1. No nível zero, o carro não possui nenhuma automação e necessita de intervenção constante do condutor. Já no nível L1, o condutor recebe informações de assistência na direção, porém este continua responsável pelo controle do veículo. Por outro lado, veículos parcialmente automatizados realizam o controle da direção e da aceleração sem intervenção humana para evitar acidentes. No nível 3, o veículo realiza mais de forma independente, como a coleta de informações sobre o ambiente, sendo o controle do veículo assistido pelo condutor, que deve permanecer atento a eventuais falhas ao longo de toda a viagem, pois o controle pode ser revertido ao usuário. Veículos com classificação L4 estão um nível abaixo dos veículos completamente automatizados, necessitando apenas da observação do usuário no controle do veículo. Por fim, os veículos L5 dispensam qualquer intervenção humana para o seu funcionamento, podendo ser supervisionados remotamente.

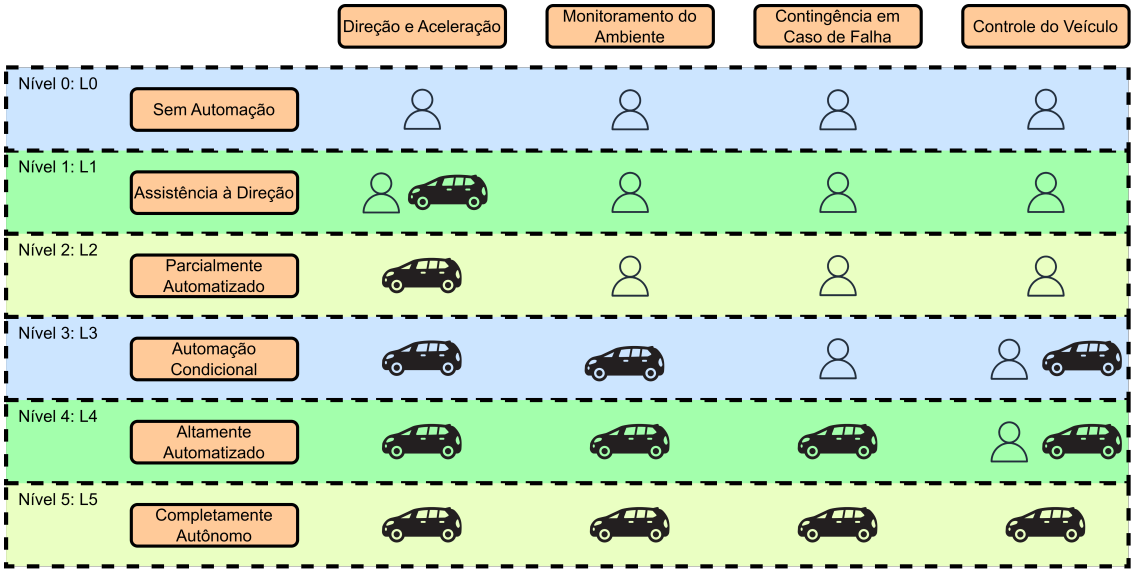


Figura 2.1. Classificação de veículos autônomos de acordo com as tarefas executadas. O ícone de usuário indica que a atividade necessita de intervenção humana, o veículo com o usuário indica que a tarefa é parcialmente automatizada e necessita de supervisão, enquanto o veículo indica que a tarefa é completamente automatizada.

O pilar dessas inovações reside em modelos de inteligência artificial, que exigem um treinamento com dados para oferecer alto desempenho em funções críticas, como a predição de situações de risco futuras e o controle automatizado de parâmetros internos do automóvel. Diante da necessidade de utilizar dados reais e pessoais do usuário para o aperfeiçoamento dessas tecnologias, surge o desafio de proteger a segurança do condutor, o que torna o uso do aprendizado federado (*Federated Learning - FL*) [McMahan et al., 2017] uma solução indispensável. Essa abordagem permite que o treinamento dos modelos de inteligência artificial ocorra de forma eficiente, evitando a transferência direta de dados sensíveis dos usuários para centros de processamento externos, garantindo assim a privacidade das informações coletadas pelo veículo. Além disso, o compartilhamento de modelos, em vez de dados, reduz a quantidade de informação

transmitida, considerando que os clientes estão constantemente coletando novas amostras de alta dimensionalidade.

Tradicionalmente, o FL utiliza o FedAvg [McMahan et al., 2017], um algoritmo de treinamento federado que atualiza o modelo global com a média ponderada dos pesos do modelo de diferentes clientes, ponderada pela quantidade de dados de cada cliente. Cada rodada de atualização é realizada de forma síncrona, na qual o servidor aguarda a resposta de todos os clientes selecionados por um determinado tempo limite para atualizar o modelo global com as respostas recebidas. A fim de evitar sobrecarga computacional para os clientes, os autores propõem selecionar uma porcentagem para atualizar o modelo em cada rodada. No entanto, o aprendizado federado apresenta desafios, como a dificuldade de convergência devido à heterogeneidade dos dados dos clientes [McMahan et al., 2017, Li et al., 2020]. Como o modelo global é treinado indiretamente, quando os clientes têm distribuições de dados distintas, as atualizações do modelo tendem a ser discordantes dependendo de quais clientes participam do treinamento. Portanto, a seleção do subconjunto de clientes que participa de uma rodada de treinamento pode impactar a qualidade do modelo treinado [Kairouz et al., 2021].

Em um cenário veicular onde os clientes estão em constante movimento, esse desafio é agravado pela heterogeneidade do canal dos clientes [Bao et al., 2021, Buyukates e Ulukus, 2021, de Souza et al., 2025], que pode gerar longos períodos de desconexão ou falhas de comunicação. Além disso, os dispositivos em um ambiente veicular são heterogêneos [EletronicsMaker, 2024] e a suposição de que o treinamento deve ser realizado em um modelo comum para o cálculo da média torna-se irrealista. Ainda, desafios práticos de aplicações veiculares, como a falta de rótulos para o treinamento de modelos de detecção de objetos, tornam o cenário mais desafiador [Li et al., 2022a].

O objetivo principal do minicurso é contextualizar a execução básica e a relevância do aprendizado federado veicular (*Vehicular Federated Learning* – VFL), apresentando como o arcabouço do VFL permite o treinamento de modelos de inteligência artificial, garantindo a privacidade dos dados dos usuários. Dessa forma, introduz-se a arquitetura do VFL, detalhando estratégias de agregação, como as abordagens centralizadas, hierárquicas e descentralizadas, para superar os desafios de latência e mobilidade. Além disso, abordam-se os esquemas de coordenação semissíncronos [Wu et al., 2021a] e assíncronos [Xie et al., 2019], que buscam adequar o FL a cenários dinâmicos, relaxando a necessidade de sincronismo e evitando problemas de atraso.

Outro objetivo é discutir a infraestrutura de rede necessária para o VFL, utilizando uma abordagem de divisão em planos de dados e controle para otimizar a transmissão de modelos e a orquestração de tarefas. Para isso, comparam-se tecnologias de transmissão física como 5G & *Beyond*, enlaces laterais e satélites, além de protocolos de aplicação (MQTT, HTTP), transporte (TCP, UDP) e rede (IP, ICN). O curso também identifica e oferece caminhos para a superação de desafios práticos, incluindo a heterogeneidade de hardware dos clientes, as limitações de largura de banda do canal sem fio, a distribuição estatística de dados entre os veículos e a escassez de dados rotulados.

Para consolidar o conhecimento teórico, o minicurso capacita os participantes por meio de uma atividade prática de simulação realista de um ambiente veicular. A simulação inclui a definição de heterogeneidade de dados, dispositivos, mobilidade e o canal de

comunicação. Ainda, os dados gerados são integrados ao arcabouço Flower para execução do aprendizado federado veicular de forma realista, avaliando o impacto de diferentes parâmetros no desempenho do modelo.

Finalmente, o curso pretende motivar o desenvolvimento de pesquisas sobre tendências futuras, como a personalização de modelos e a destilação de conhecimento para arquiteturas heterogêneas, visando a implantação eficiente de soluções inteligentes em ambientes de alta mobilidade.

2.2. Aprendizado Federado e seu uso em Redes Veiculares

Tradicionalmente, o Aprendizado de Máquina (*Machine Learning – ML*) requer que um conjunto de dados representativo seja reunido em um único local, o que representa um desafio para redes veiculares, dado o volume de dados produzido por cada veículo e a sensibilidade inerente a esses dados [Chellapandi et al., 2023, Zhang et al., 2023]. Por exemplo, foi estimado que Veículos Conectados e Automatizados (*Connected and Autonomous Vehicles – CAV*) produzem entre 20 e 40 TB de dados por dia [Chellapandi et al., 2023], contendo informações que permitem a caracterização dos veículos e dos condutores. Alternativamente, o FL é um paradigma de Aprendizado Descentralizado de Máquina (*Decentralized Machine Learning – DML*) em que um conjunto de clientes, por exemplo, dispositivos móveis, executa o aprendizado colaborativo de um modelo sem compartilhar seus dados [McMahan et al., 2017]. Assim, cada cliente possui uma cópia local do modelo de ML a ser ajustado, sobre o qual aplica modificações com base em seus dados privados. Periodicamente, os clientes compartilham as atualizações feitas aos modelos locais para que sejam agregadas pelo servidor. Por sua vez, o servidor combina as atualizações dos clientes em um modelo global, que, posteriormente, é difundido aos clientes da federação, para que o processo de aprendizado seja repetido. O ciclo de uma atualização do modelo global é denominado rodada, enquanto as atualizações locais dos clientes são denominadas épocas locais. Sob essa estrutura de aprendizado, o FL permite o treinamento de modelos, mitigando problemas que surgem em um aprendizado exclusivamente local, como a baixa representatividade do conjunto de dados disponível em relação à distribuição real do fenômeno a ser aprendido [McMahan et al., 2017, Yuan et al., 2024].

Embora o FL aborde o problema da privacidade de dados e a redução dos custos de comunicação associados ao ML em redes veiculares [Posner et al., 2021], sua configuração tradicional implica hipóteses e restrições que podem dificultar seu emprego efetivo nessas redes. Um dos principais desafios à implementação de FL em redes veiculares é a alta mobilidade dos clientes. A federação pode sofrer modificações decorrentes da inclusão e da remoção de clientes, uma vez que veículos podem entrar e sair da área de cobertura do servidor de agregação [Fittipaldi et al., 2025]. Além disso, as informações enviadas pelos membros da federação podem chegar com atrasos distintos devido às condições variáveis do canal de comunicação e às distâncias entre clientes e agregadores [Zhang et al., 2023]. Por fim, diferenças no ambiente em que cada cliente federado se encontra, bem como na qualidade dos equipamentos de coleta empregados, provocam heterogeneidade nos dados coletados [Chellapandi et al., 2023]. Por consequência, as estratégias de agregação de informação e de coordenação dos clientes veiculares devem considerar as características de cada rede veicular, a fim de garantir o sucesso do processo

de aprendizado. No que segue, descreveremos sistematicamente as principais estratégias de agregação e de coordenação de clientes federados.

2.2.1. Estratégias de agregação

No FL clássico, as informações aprendidas pelos clientes, como os parâmetros ou gradientes do modelo, são enviadas a um único nó-servidor, responsável por agregá-las, difundir os parâmetros do modelo global e orquestrar a federação [McMahan et al., 2017]. Devido à centralização da etapa de agregação, essa variação é conhecida como Aprendizado Federado Centralizado (*Centralized Federated Learning – CFL*) [Beltrán et al., 2023], e o nó-agregador é referido como servidor de agregação. Essa centralização traz alguns problemas que a tornam inadequada a alguns cenários encontrados em redes sem fio [Beltrán et al., 2023, Yuan et al., 2024]. O primeiro deles é a existência de um único ponto de falha, que pode interromper o processo de aprendizado quando o servidor fica incomunicável ou torna-se inoperante [Li et al., 2021b, Yuan et al., 2024]. Adicionalmente, a largura de banda do servidor pode ser um gargalo quando múltiplos clientes enviam atualizações simultaneamente, retardando a operação de agregação, e as distâncias entre clientes e servidor podem aumentar os custos de comunicação [Posner et al., 2021, Chen et al., 2023a]. Além disso, centralizar a operação de agregação requer a confiança dos clientes no servidor da federação [Kalra et al., 2023], uma vez que servidores maliciosos podem extrair informações sensíveis dos dados recebidos [Huang et al., 2021].

O Aprendizado Federado Descentralizado (*Decentralized Federated Learning – DFL*), também conhecido como FL Distribuído ou sem servidor (*Serverless FL*) [Beltrán et al., 2023], é uma variação em que a etapa de agregação ocorre de maneira distribuída nos próprios clientes [Beltrán et al., 2023, Yuan et al., 2024]. No DFL, os clientes federados compartilham suas atualizações diretamente com seus parceiros, por meio de comunicação ponto a ponto (*Peer-to-Peer* – P2P), difusão (*broadcast*) ou fofoca (*gossiping*) [Yuan et al., 2024]. Isso traz duas vantagens em relação ao CFL. Uma comunicação mais direta entre nós permite reduzir os custos associados à transmissão de dados e evita que muitos clientes se comuniquem simultaneamente com um único nó [Chen et al., 2023a, Yuan et al., 2024]. Por fim, o DFL permite uma maior personalização de modelos do que o CFL, pois os próprios clientes federados podem decidir com quem interagir durante o processo de aprendizado e como agregar as atualizações recebidas [Jeong e Kountouris, 2023, Liu et al., 2024b].

Se, por um lado, o DFL permite a remoção de um único ponto de falha e maior liberdade para definir como os clientes interagem entre si, essa variação pode apresentar desvantagens em relação ao CFL quanto ao tempo de convergência e à qualidade dos modelos aprendidos. A topologia da federação, que diz respeito a com quem cada cliente federado se comunica durante o processo de aprendizado, desempenha um papel fundamental na taxa de convergência da federação e na capacidade de generalização do modelo aprendido [Shi et al., 2023, Sun et al., 2025]. Isso ocorre porque os dados de veículos próximos seguem uma distribuição semelhante, que pode divergir da distribuição real. Logo, o modelo resultante de sua interação pode sobreajustar-se a essa tendência. Além disso, agregar modelos muito distintos reduz o desempenho e aumenta o número de iterações até a convergência do modelo [Shi et al., 2023]. Topologias densas, nas quais clientes que

participam de DFL estabelecem conexões com um grande número de membros da federação, tendem a apresentar modelos que generalizam melhor. Entretanto, essa configuração pode acarretar altos custos de comunicação [Sun et al., 2025], particularmente em redes de grande escala.

Considerando as vantagens e desvantagens de CFL e DFL, o Aprendizado Federado Híbrido ou Semi-DFL (SDFL) apresenta-se como um conjunto de estratégias de agregação que combina características de ambas as abordagens anteriores [Beltrán et al., 2023]. Dentre essas estratégias, uma das principais é o Aprendizado Federado Hierárquico (*Hierarchical Federated Learning –HFL)*) [Abad et al., 2020, Liu et al., 2020]. Nessa variação, constrói-se uma hierarquia de nós agregadores, na qual clientes que formam a base da hierarquia se organizam em pequenos aglomerados (*clusters*) e geram um modelo do nível hierárquico. Os modelos são agregados em múltiplas etapas, seguindo a estrutura hierárquica [De Rango et al., 2021, Azimi-Abarghouyi e Fischione, 2025].

Assim, o HFL intercala momentos de agregação intra-aglomerado (*intra-cluster*), semelhante ao CFL clássico, com momentos de agregação inter-aglomerado para estabelecer um modelo global [Abad et al., 2020]. Em redes veiculares, estações radio-base ou Unidades de Borda de Estrada (*Roadside Units* – RSU) podem ser exploradas como servidores de agregação locais próximos aos veículos, cujos modelos são enviados a um servidor em nuvem, para gerar o modelo global [Xie et al., 2026]. Outra variação, mais próxima do DFL, consiste em eleger periodicamente um dos clientes como servidor de agregação [Beltrán et al., 2023]. Essa abordagem permite reduzir a necessidade de infraestrutura dedicada à agregação na borda, além de mitigar a sobrecarga persistente em um único nó agregador.

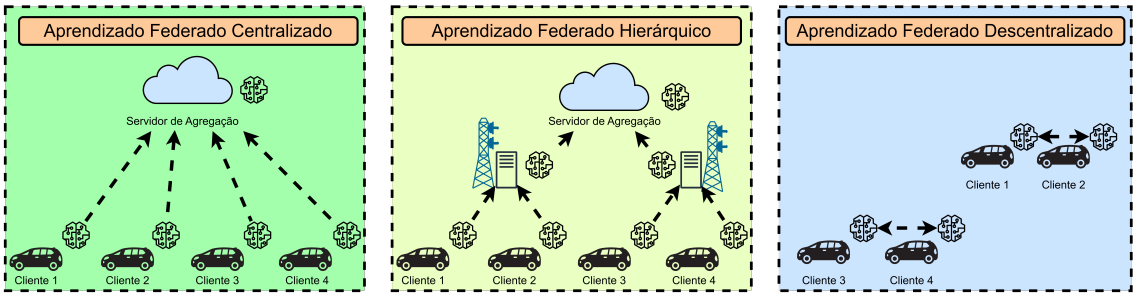


Figura 2.2. Comparação entre as diferentes estratégias de agregação.

De modo geral, pode-se considerar o CFL e o DFL como dois extremos de estratégias de agregação, e as demais variações combinam e privilegiam características de ambos, conforme os requisitos operacionais de cada implementação de FL, como mostrado na Figura 2.2. Outro fator importante para o sucesso do FL é definir quando a agregação deve ocorrer, o que é abordado pelas diferentes estratégias de coordenação.

2.2.2. Estratégias de coordenação

Enquanto as estratégias de agregação definem o local de agregação dos modelos, as estratégias de coordenação definem o instante em que os modelos serão agregados. Como no caso anterior, em que temos um conjunto de estratégias cujos extremos são CFL

e DFL, pode-se classificar as estratégias de coordenação em síncronas, semissíncronas e assíncronas. O FL clássico emprega uma estratégia de coordenação em que a operação de agregação ocorre somente quando o servidor recebe as atualizações de todos os clientes, após esses executarem o mesmo número de épocas locais até um tempo limite (*timeout*) [McMahan et al., 2017]. Isso implica que, a cada operação de agregação, todos os clientes devem estar sincronizados na mesma rodada e que o servidor aguarde que todos encerrem seus processos de aprendizado locais e transmitam suas atualizações. A exceção são os clientes que eventualmente estejam desconectados ou em um estado de falha.

Essas premissas podem ser problemáticas quando a federação é composta por dispositivos heterogêneos e a transmissão de atualizações ocorre por meio de redes sem fio. As diferenças de recursos entre dispositivos podem provocar variações no tempo de execução dessas operações locais, e clientes com uma conexão ruim com o servidor de agregação impactam negativamente o tempo de início da operação de agregação [Wang et al., 2022]. Em um FL síncrono, como descrito, o servidor e os dispositivos que executam o aprendizado local rapidamente devem aguardar os dispositivos retardatários (*stragglers*). Isso impacta o tempo de convergência da federação e torna o aprendizado ineficiente devido a ociosidade dos clientes com mais recursos computacionais [Xie et al., 2019, Xu et al., 2023].

Para mitigar o efeito de clientes retardatários, pode-se relaxar o requisito que obriga o servidor a esperar todas as atualizações dos clientes da federação, permitindo que esse agregue modelos treinados em rodadas diferentes, após receber um número fixo de atualizações, ou ainda sempre que recebe uma nova atualização. O último desses é conhecido como FL assíncrono [Xie et al., 2019], enquanto os dois primeiros são geralmente classificados como estratégias de FL semissíncrono [Stripelis et al., 2022, Nguyen et al., 2022].

No FL assíncrono, procura-se minimizar ao máximo os estados ociosos, gerando um novo modelo global a cada atualização, eliminando completamente o atraso decorrente da espera aos clientes retardatários. Entretanto, o emprego efetivo do FL assíncrono requer atenção quanto ao algoritmo de agregação. Uma vez que clientes em uma mesma federação utilizam versões distintas do modelo global, há um impacto negativo tanto a taxa de convergência quanto na qualidade do modelo aprendido. Em particular, clientes retardatários possuem modelos obsoletos que degradam o modelo global se agregados com o mesmo peso que os demais [Liu et al., 2024a].

Dessa forma, o FL semissíncrono propõe uma solução intermediária em que a operação de agregação ocorre de forma semelhante ao FL síncrono, porém elimina a restrição de aguardar que todos os clientes enviem suas atualizações [Stripelis et al., 2022]. Para mitigar a obsolescência dos modelos locais, o FL semissíncrono define momentos de sincronização entre um número mínimo de clientes. Porém, clientes retardatários não têm seus modelos descartados, como ocorre no caso síncrono. Caso o servidor receba um modelo de um cliente que chegue após a agregação, deve-se determinar o atraso em relação ao modelo global. O objetivo é decidir se o modelo defasado pode contribuir com o modelo ou se deve ser descartado para preservar o desempenho do modelo global. O grau de defasagem tolerável e o peso na agregação dos modelos defasados são hiperparâmetros

do FL semissíncrono. Entretanto, o FL semissíncrono também está sujeito a problemas semelhantes aos do FL assíncrono, como o problema de convergência [Islam et al., 2025].

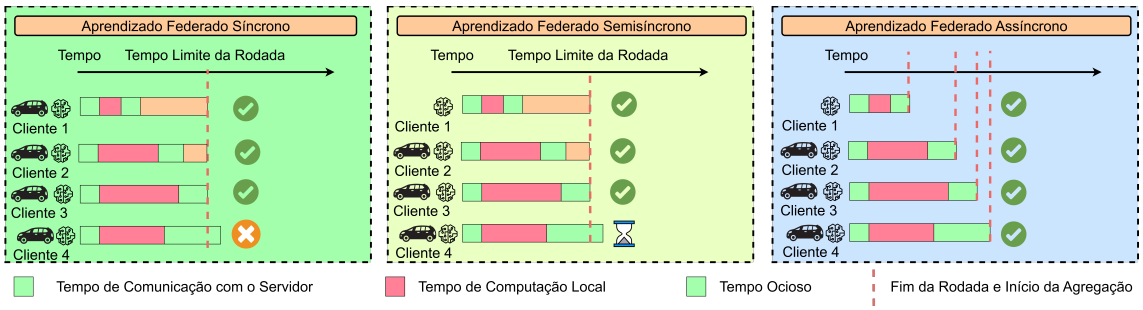


Figura 2.3. Comparação entre as diferentes estratégias de coordenação.

A Figura 2.3 apresenta as diferenças entre as principais estratégias de coordenação. Cada estratégia de coordenação possui vantagens e desvantagens, relacionadas às diferentes tecnologias de comunicação dos clientes para a realização do treinamento. Assim, a seguir, abordam-se as principais características da infraestrutura, detalhando os principais protocolos utilizados nas camadas inferiores da arquitetura.

2.3. Infraestrutura da Rede do Aprendizado Federado Veicular

A infraestrutura da rede do FL no cenário veicular é formada por componentes de *hardware* e *software*, cuja implantação é dividida em três setores: borda veicular, núcleo e a computação em nuvem. A borda veicular inclui os dispositivos nos quais os dados são gerados e os dispositivos com as quais eles se comunicam diretamente por um enlace. O núcleo inclui toda a infraestrutura tradicional de rede utilizada para interconectar as redes de acesso na borda aos servidores em nuvem. A computação em nuvem inclui as redes de *data centers* e os servidores de alta capacidade computacional, responsáveis pelo gerenciamento do FL e pela agregação de modelos.

O projeto da infraestrutura de rede para FL veicular em todos estes setores deve levar em consideração três frentes principais, discutidas ao longo das seguintes subseções. A Subseção 2.3.1 discute a frente de tecnologias físicas de transmissão utilizadas para implementar os enlaces da borda veicular. Em seguida, a Subseção 2.3.2 apresenta a frente de protocolos de comunicação utilizados ao longo de toda a rede, que têm por objetivo a garantia de desempenho e resiliência a falhas distribuídas. Por fim, a Seção 2.3.2 foca na frente de gerenciamento de sistemas em rede com o objetivo de suportar a inicialização, a finalização, o monitoramento e a atualização das tarefas de FL.

A Tabela 2.1 elenca as principais tecnologias utilizadas em diversas camadas de rede para suporte ao FL, incluindo protocolos, padrões, arquiteturas e algoritmos. As tecnologias em verde já são adotadas em VFL, e serão discutidas ao longo das Seções 2.3.1 e 2.3.2. As tecnologias em violeta são propostas para resolver problemas emergentes em comunicações para VFL, sendo discutidas em mais detalhes na seção de desafios desse minicurso, Seção 2.4.3.

Tabela 2.1. Comparativo de protocolos e tecnologias por camada, onde a cor verde denota tecnologias já adotadas em VFL, enquanto que a cor violeta denota propostas para solucionar desafios.

Camada	Protocolos/Tecnologia	Pontos fortes	Limitações
Aplicação	HTTP+REST	Não mantém estado de conexão	Sobrecarga com URI
	CoAP	Adequado para IoT	Fragmentação e segurança
	gRPC	Resiliente a redes desafiadoras	Dependência de conexão persistente
	MQTT/AMQP	Assincronismo	Maior latência devido ao broker
Transporte	TCP	Confiabilidade e controle de congestionamento	Sobrecarga de controle e conexão
	UDP	Evita sobrecarga de controle e conexão	Perdas e congestionamento
	PCON	Controle ativo de fila de roteamento	Exige modificações na camada de rede
Rede	IP	Compatível com a infraestrutura atual da Internet	Melhor esforço (poucas garantias)
	ICN NDN	Cache, multicast, mobilidade e ECN	Camada de adaptação ao IP traz sobrecarga
Acesso ao Meio (MAC) e transmissão física	Redes celulares (4G, 5G)	Alcance, vazão e latência	Custo com plano de serviço das operadoras
	IEEE 802.X	Equipamentos de baixo custo e consumo	Baixo alcance e interferência
	Redes space-air-ground	Cobertura em certas regiões	Complexidade da infraestrutura
	Novas técnicas de DSP (NOMA)	Maior eficiência espectral	Equipamentos novos e mais caros
	RIS	Aumento do alcance com baixa energia	Necessidade de instalação física

REST: *Representational State Transfer*
URI: *Uniform Resource Identifier*
CoAP: *Constrained Application Protocol*
gRPC: *gRPC Remote Procedure Calls*
AMQP: *Advanced Message Queuing Protocol*
PCON: *Practical Congestion Control*
ICN: *Information-Centric Networking*
NDN: *Named Data Networking*
ECN: *Explicit Congestion Notification*
DSP: *Digital Signal Processing*
NOMA: *Non-Orthogonal Multiple Access*
RIS: *Reconfigurable Intelligent Surface*

2.3.1. Tecnologias físicas de transmissão e acesso

Existem muitas propostas que usam FL para otimização de sistemas de comunicação sem fio veiculares. Entretanto, a discussão nesta subseção foca em tecnologias físicas de transmissão utilizadas para suportar o FL veicular. O objetivo é criar enlaces entre os *hardwares* presentes na borda veicular, ou seja, os dispositivos nos quais os dados são gerados e outros dispositivos em borda.

Na borda veicular, os dispositivos que geram dados estão presentes nos veículos, nos *smartphones* dos pedestres e passageiros e em equipamentos IoT. A rede local dos veículos inclui desde dispositivos simples, tais como sensores, até dispositivos com maior poder computacional, como computadores de bordo e Unidades Eletrônicas de Controle (*Electronic Control Units* – ECUs). No contexto de transmissão de dados em redes veiculares, o foco é em uma dessas ECUs, chamada de ECU primária, ou *gateway*, que provê conectividade sem fio com o restante da infraestrutura [Kornaros et al., 2020]. Existem também dispositivos dedicados ao provisionamento de redes de acesso, permitindo o en-

caminhamento de mensagens para outros dispositivos nas proximidades ou para a rede de núcleo, visando alcançar servidores em nuvem. Em termos de processamento, alguns dispositivos que geram dados podem ser incapazes de treinar modelos localmente, delegando essa tarefa para unidades de borda devidamente equipadas, como as RSUs (*Road Side Units*) [Magdum et al., 2021] localizadas na proximidade.

A transmissão dos dados pode ser direta entre veículos (*Vehicle-to-Vehicle* – V2V), entre o veículo e os outros dispositivos na infraestrutura de borda (*Vehicle-to-Infrastructure* – V2I) ou entre o veículo e o restante da rede (*Vehicle-to-Network* – V2N). Todas estas abordagens fazem parte do guarda-chuva de comunicações V2X (*Vehicle-to-Everything*). A abordagem tradicional do FL veicular depende de comunicação V2N, uma vez que o servidor de agregação está em nuvem. Algumas propostas evidenciam ganhos de latência ao posicionar o servidor de agregação no ponto de acesso e utilizar comunicação V2I, se servindo do padrão em MEC (*Multi-access Edge Computing*). Por fim, as abordagens de FL descentralizado dependem da troca direta de modelos entre os veículos por meio de V2V [Zejun et al., 2024].

A comunicação sem fio entre estes dispositivos tipicamente utiliza padrões comuns em redes celulares, como o 4G (*Long-Term Evolution* – LTE) e o 5G (*New Radio* – NR). Esses padrões, definidos pela 3GPP (*3rd Generation Partnership Project*) usam tecnologias de acesso por rádio com alta vazão e baixa latência, necessárias para a transmissão de modelos grandes no FL veicular. O acesso ao meio compartilhado e a modulação utilizam o OFDMA (*Orthogonal Frequency Division Multiple Access*), no qual cada usuário recebe um bloco de recurso (*Resource Block* – RB), que consiste em uma faixa de frequência e *slot* de tempo no qual ocorre a transmissão do sinal modulado com os dados. Alguns *slots* são dedicados para *uplink* (equipamento do usuário para estação base) e outros são dedicados para *downlink* (estação base para equipamento do usuário). A implantação das redes de acesso pro rádio já está bastante madura, garantindo uma cobertura com ampla penetração no território urbano [Magdum et al., 2021].

As redes de quarta e quinta geração incluem padrões para comunicação direta entre equipamentos do usuário (*User Equipments* – UE). Entretanto, a maioria das implantações é configurada para comunicação direta com as estações rádio base, desmotivando o uso de enlaces laterais (*side-links*). Ainda assim, a inclusão de padrões mais específicos de V2X pode reduzir a latência de sistemas de FL que exigem comunicação entre componentes em borda. Outras tecnologias promissoras na camada física incluem o padrão IEEE 802.11p para V2V, permitindo também comunicações diretas de curto alcance (*Dedicated Short-Range Communications* – DSRC) [Magdum et al., 2021].

2.3.2. Protocolos de rede e sistemas distribuídos

Na maioria das propostas de FL veicular, a troca de parâmetros entre servidor e clientes ocorre através de mensagens que trafegam pela Internet. Isso se deve ao fato de que a rede de acesso inclui apenas os dispositivos na borda veicular, enquanto que o servidor de agregação é uma máquina em nuvem. Nesses cenários, além das tecnologias físicas de transmissão sem fio, outra frente que deve ser considerada no projeto da infraestrutura de rede do FL veicular é a escolha dos protocolos. Para estudar as possibilidades de protocolos para FL veicular, será considerada a pilha de protocolos da arquitetura em

camadas da Internet (TCP/IP), a menos que dito o contrário. O foco será nas camadas superiores (aplicação e transporte) A camada de rede utiliza o IP para encaminhar os pacotes entre redes de forma interoperável, segundo o modelo de ampulheta (*cintura fina*) da Internet. As camadas de enlace e física são discutidas em outras seções.

A camada de aplicação é a mais flexível a nível de projeto, uma vez que o desenvolvedor tem a liberdade de construir suas próprias máquinas de estados finitas (*Finite-State Machine* – FSM) no nível de *software* de aplicação. A FSM define quais mensagens serão enviadas, quando serão enviadas e o que será feito quando elas forem recebidas, tanto no cliente quanto no servidor. Em geral, essa troca de mensagens nas aplicações envolve o *download* de parâmetros no início de uma rodada, o *upload* de parâmetros no final da rodada e mensagens de controle utilizadas pelos clientes para ingressar em uma federação (no início) e sincronizar o seu estado com o servidor (periodicamente). Com um projeto adequado de protocolo de aplicação, é possível resolver problemas típicos de sistemas distribuídos, como tolerância a falhas e garantia de sincronismo.

Outro aspecto importante no desenvolvimento da aplicação é o formato das mensagens trocadas entre nós no FL. Heydarishirayeh implementa uma ferramenta de FL que permite a escolha de diferentes protocolos de aplicação tradicionais para troca de mensagens [Heydarishirayeh, 2023]. Os experimentos utilizaram quatro opções (REST+HTTP, CoAP, gRPC+HTTP2 e MQTT/AMQP), a serem resumidas a seguir. Os autores introduziram condições de rede desafiadoras, simulando cenários com muitas perdas, baixa vazão e alta latência. Esse cenário é comum nos enlaces sem fio da rede de acesso em certos locais com sombras e interferência.

O paradigma REST (*Representational State Transfer*) permite a criação de APIs *web* sem estado (*stateless*) utilizando mensagens HTTP, sendo usada na ferramenta de FL FLoBC [Ghanem et al., 2022]. Entretanto, essa abordagem introduz uma sobrecarga significativa em termos de *bytes* de controle devido ao fato de que todas as mensagens incluem um cabeçalho com URI (*Uniform Resource Identifier*). O CoAP (*Constrained Application Protocol*) é uma alternativa baseada em REST voltada para dispositivos IoT leves, mas que apresenta alta latência em condições de rede desafiadoras. A necessidade de fragmentação em pacotes menores que 1 kB, a falta de recursos de segurança e a possibilidade de aumento do congestionamento pelo uso de UDP desestimulam o uso desse protocolo em aplicações reais. O gRPC (Google *Repote Procedure Call*) permite com que um cliente chame uma função em um servidor, abstraindo em uma biblioteca, chamada *stub*, a formatação dos parâmetros e da resposta em XML e o envio da mensagem por HTTP2. Essa é a abordagem utilizada pelas ferramentas mais populares: Flower e TensorFlow Federated (TFF). Nos experimentos realizados em [Heydarishirayeh, 2023], esse protocolo foi o menos afetado por condições de rede desafiadoras. Por fim, o MQTT (*Message Queuing Telemetry Transport*) e o AMQP (*Advanced Message Queuing Protocol*) são protocolos que seguem o paradigma *publish-subscribe*, no qual consumidores se inscrevem em filas de mensagens para receber mensagens de produtores. O gerenciamento de filas, feito por um *broker*, garante assincronia na transferência de mensagens, motivando o seu uso na ferramenta FedML [He et al., 2020]. Entretanto, a implantação da infraestrutura de rede nesse cenário é mais complexa e introduz maior latência devido à dependência do intermediário.

Na camada de transporte, o TCP é preferível na maior parte dos casos devido à criação de um canal de comunicação confiável e ao controle de congestionamento. Os protocolos de aplicação baseados em MQTT/AMQP e HTTP se servem de conexões TCP, apesar de ser possível também o uso direto de soquetes TCP sem esses protocolos previamente citados. As implementações mais comuns do TCP são o BIC e o CUBIC, utilizadas no *kernel* do Linux. Entretanto, Amadeo *et al.* indicam o uso do TCP Westwood para melhorar a conexão dos enlaces sem fio do FL e do *multipaht* TCP (MPTCP) para evitar congestionamento em pontos específicos da rede de núcleo quando múltiplos clientes enviam parâmetros concorrentemente [Amadeo et al., 2025]. Cleland *et al.* demonstram que, apesar da menor sobrecarga de controle, o UDP não reduz a latência na transferência de modelos em relação ao TCP no FL em redes de borda 4G e 5G [Cleland et al., 2022]. Apesar do TCP desempenhar pior em redes mais limitadas (3G), a ausência de mecanismos de confirmação e retransmissão inviabiliza a convergência do modelo com o uso do UDP nessas condições.

2.3.3. Gerenciamento do aprendizado federado

Embora no ML centralizado típico os profissionais possam facilmente explorar diferentes modelos e hiperparâmetros, no FL, os operadores na nuvem não têm controle sobre os dispositivos onde o treinamento é realizado. Portanto, o desenvolvimento de arquiteturas para automatizar a criação, implantação, modificação e monitoramento de tarefas de FL é desejável. As operações de FL incluem o desenvolvimento de código, implantação em clientes compatíveis selecionados, monitoramento e tomada de decisão automática ou manual.

Alguns trabalhos na literatura propõem ferramentas para reduzir a sobrecarga de gerenciamento humano. Wang *et al.* desenvolvem o FLINT (FL *Integration*), que aborda os desafios relacionados à seleção de clientes com limitação de recursos heterogêneos e disponibilidade intermitente, considerando uma única tarefa [Wang et al., 2023]. Moon *et al.* implementam o FedOps para gerenciar o ciclo CI/CD (*Continuous Integration and Delivery*) dos modelos de FL resultantes [Moon et al., 2024].

Quebrar um código monolítico em microsserviços, tipicamente implantados como contêineres, facilita modificações e manutenção. Alguns autores adotaram esse conceito para construir plataformas de FL. O Micro-FL, por exemplo, melhora a escalabilidade e a tolerância a falhas de uma única tarefa de FL usando Docker e Kubernetes [Sabuhi et al., 2024]. Além disso, o FLIP (FL *Interactive Platform*) adiciona camadas sobre o arcabouço Flower para implantar tarefas como contêineres Docker, assumindo que o nó de gerenciamento tenha acesso ininterrupto aos clientes [Galende et al., 2024].

As implantações de FL são altamente dinâmicas, uma vez que: i) novos clientes podem surgir, e ii) as estratégias de treinamento podem ser alteradas pelo especialista, justificando trabalhos sobre a transferência *Over-the-Air* (OTA) do código do cliente. Huedo *et al.* distribuem máquinas virtuais para clientes de FL em infraestruturas de borda geodistribuídas, sem considerar a necessidade de atualizações em tempo real [Huedo et al., 2025]. Chahoud *et al.* implantam novos clientes em contêineres Docker *on-the-fly* [Chahoud et al., 2023a]. Como selecionar os clientes mais adequados pode ser uma tarefa muito difícil, os autores também propuseram o uso de Algoritmos Genéti-

cos e Aprendizado por Reforço [Chahoud et al., 2023b, Chahoud et al., 2025]. O FLScalize, de Yang *et al.*, usa o GitHub para aplicar modificações nos códigos do cliente e do servidor durante a implantação contínua [Yang et al., 2023].

Thomaz *et al.* apresentam a AGATA, uma arquitetura para automatizar o provisionamento e a modificação de tarefas de FL para clientes apropriados, assumindo um cenário de CI/CD. Isso permite *pipelines* semelhantes aos geralmente encontrados em implantações centralizadas, nos quais os cientistas de dados focam apenas em aprimorar seus modelos, abstraindo a distribuição do aprendizado entre os dispositivos participantes. Além disso, a sobrecarga para executar as operações de gerenciamento na AGATA é muito menor do que a sobrecarga introduzida pela própria execução do FL [Thomaz et al., 2025].

2.4. Desafios do Aprendizado Federado Veicular

O FL tradicional possui desafios inerentes ao sistema distribuído, como a heterogeneidade dos dispositivos e dados. Para um cenário de aprendizado síncrono, a heterogeneidade de dispositivos implica a espera de clientes mais rápidos por clientes retardatários, o que prolonga a duração do treinamento do modelo global. Por outro lado, a presença de distribuições heterogêneas nos clientes faz com que os pesos gerados ao longo do treinamento tendam a divergir. Além dos desafios presentes no FL tradicional, o VFL introduz desafios devido ao alto grau de mobilidade dos clientes e às necessidades da aplicação veicular. Assim, nesta seção são inicialmente discutidos os desafios do FL em geral, devido à heterogeneidade de dispositivos e dados. Após isso, apresentam-se os desafios específicos do VFL, discutindo a heterogeneidade no canal de comunicação dos clientes e desafios específicos de aplicações veiculares.

2.4.1. Heterogeneidade de dispositivos

Um dos principais desafios para os sistemas distribuídos é assegurar a evolução do desempenho ao longo do tempo de forma justa quando os clientes possuem dispositivos heterogêneos. Em um cenário veicular os dispositivos apresentam uma alta variação de capacidade computacional. Enquanto veículos autônomos de nível 2 possuem dispositivos capazes de realizar 2,5 teraoperações por segundo (TOPS), veículos de nível 5 realizam mais de 2.000 TOPS [EletronicsMaker, 2024].

Um desafio que a heterogeneidade de dispositivo impõe ao aprendizado federado é a seleção do modelo. Dispositivos com menor poder computacional necessitam de modelos mais leves, enquanto dispositivos mais potentes podem executar modelos mais complexos. Entretanto, o algoritmo de aprendizado federado mais popular, o FedAvg, assume que os modelos são idênticos, para realizar a média dos parâmetros dos clientes no servidor.

Utilizar modelos mais complexos no sistema permite aos clientes mais potentes atingirem um alto desempenho, limitando a participação no treinamento dos clientes com poucos recursos computacionais. Por outro lado, utilizar modelos mais simples para todos os clientes subutiliza os recursos computacionais dos clientes mais potentes, além de prejudicar o desempenho do modelo. Assim, é necessária a utilização de alternativas adequadas ao uso de modelos heterogêneos de acordo com as diferentes capacidades computacionais encontradas em ambientes veiculares. Dessa forma, os clientes podem

contribuir mutuamente ao longo do treinamento, porém utilizando modelos adequados às capacidades de seus dispositivos.

Além disso, clientes com maior poder de computação finalizam sua tarefa em um tempo menor do que os clientes com menor poder computacional, ou retardatários (*stragglers*). Dependendo da disparidade entre os dispositivos, clientes mais rápidos podem aguardar dezenas de minutos entre o fim de uma rodada e o início da próxima [Kang et al., 2025].

Assim, uma forma de mitigar este desafio é impor um tempo limite de atualização. Dessa forma, clientes que ultrapassem o tempo de atualização de uma roda têm suas atualizações descartadas e o treinamento continua sem a sua interferência. No entanto, definir o tempo limite é uma tarefa árdua, pois ao reduzir o tempo limite, menos parâmetros de clientes são utilizados para atualizar o modelo global, reduzindo a velocidade de convergência. Ao mesmo tempo, aumentar o tempo limite implica aguardar por mais tempo os clientes retardatários [Ali et al., 2025].

Essa é uma motivação para o surgimento das estratégias de coordenação alternativas apresentadas na Seção 2.2.2. Ao reduzir ou eliminar a necessidade de sincronismo entre os clientes, estratégias de coordenação como o aprendizado federado semissíncrono e o assíncrono se adéquam a um ambiente com dispositivos heterogêneos [Xu et al., 2023, Wang et al., 2022, Stripelis et al., 2022, Wu et al., 2021a].

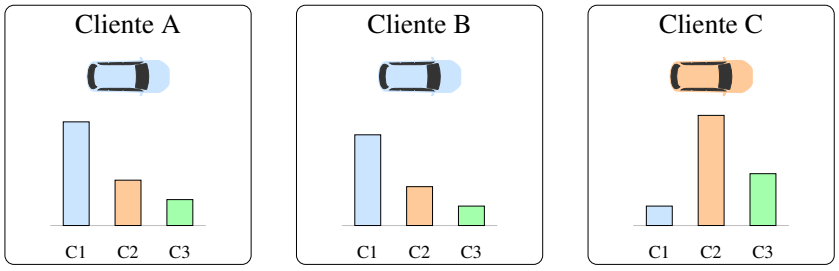
Entretanto, reduzir a necessidade de sincronismo pode penalizar clientes com maior poder computacional, uma vez que estes passam a executar mais atualizações do que clientes mais lentos. Um segundo problema mais grave é o fato de que a capacidade computacional do cliente não implica uma maior qualidade dos dados de treinamento. Assim, ao forçar que clientes com maior poder computacional realizem mais atualizações, não garante que o treinamento seja mais curto ou mesmo que o modelo treinado seja o melhor, principalmente quando os clientes possuem distribuições de dados heterogêneas [Ali et al., 2025].

2.4.2. Distribuições de dados heterogêneas

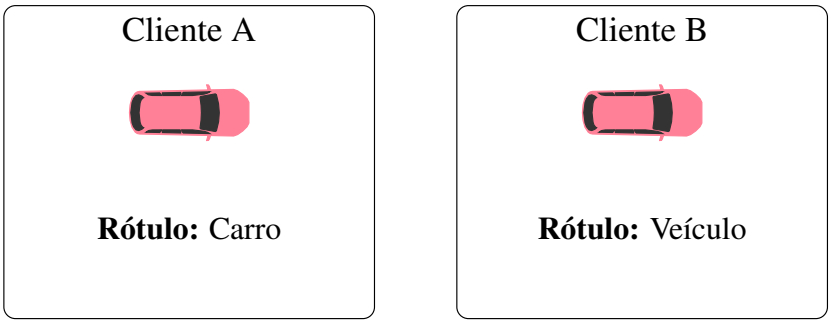
Trabalhos clássicos no contexto de FL consideram um cenário ideal, em que características dos clientes participantes no treinamento federado são distribuídas de forma homogênea. Embora avaliações nessas condições tenham demonstrado a capacidade das técnicas de FL em obter bom desempenho, esse cenário pode não representar bem casos de aplicações realistas de larga escala [Ye e ohters, 2023]. No contexto do VFL, por exemplo, a coleta dos dados não é feita de maneira uniforme. Cada veículo está inserido em um ambiente particular, com características distintas que serão representadas nos dados coletados. Um motorista que dirige no período noturno tenderá a coletar mais imagens com baixa luminosidade do que outro que apenas realiza coleta durante o dia. Se os conjuntos de dados locais dos clientes são inconsistentes e não representam bem a distribuição geral dos dados, diz-se que esses são não independentes e identicamente distribuídos (*Independent and Identically Distributed* – IID), ou não-IID [Li et al., 2022a]. Nesse contexto, Ye *et al.* propõem uma categorização das diferentes formas de heterogeneidade presentes em ambientes federados, destacando casos específicos de fontes de inconsistências entre os dados dos clientes [Ye e ohters, 2023]. A seguir, são apresentadas

essas categorias.

- **Viés de rotulação (*label skew*)**: Ocorre quando as distribuições de classes dos clientes participantes são inconsistentes. Pode se manifestar como um viés de distribuição de rótulos (*label distribution skew*) [Zhang et al., 2022b], em que o processo de coleta condiciona o conjunto de dados a ter uma representação desbalanceada das classes, ou um viés de preferência de rótulos (*label preference skew*) [Kairouz et al., 2021], em que, mesmo para dados com características consistentes entre os clientes, rótulos diferentes podem ser atribuídos em cada caso. A Figura 2.4 ilustra os dois tipos de vieses de rotulação.



(a) Diagrama ilustrativo do fenômeno de viés de distribuição de rótulos.



(b) Diagrama ilustrativo do fenômeno de viés de preferência de rótulos.

Figura 2.4. Diagramas exemplificando cenários de viés de rotulação.

- **Viés de características (*feature skew*)**: Ocorre quando as distribuições das características dos dados são inconsistentes entre os clientes. No contexto veicular, um possível exemplo de viés de características seria a diferença entre imagens de carros capturadas durante um inverno intenso com geadas e durante o verão. Nessas condições, seriam produzidas imagens com características distintas, mas que representam a mesma classe de objetos, caracterizando uma modalidade de viés de características, denominada viés de distribuição de características (*feature distribution skew*). Outra possibilidade corresponde ao viés de condição de características (*feature condition skew*), que ocorre quando, mesmo considerando uma mesma classe de interesse, diferentes clientes observam subconjuntos distintos. Enquanto um cliente que habita um cenário urbano pode observar predominantemente veículos leves como carros, clientes de ambientes rurais podem coletar imagens de caminhões. Mesmo que se atribua a mesma classe "Veículo" para ambos os casos, ocorrerá uma inconsistência entre as características dos mesmos.

- **Viés de qualidade (*quality skew*)**: Representa inconsistência entre a qualidade da instrumentação de captura de dados e do procedimento de rotulação entre os clientes. Pode se manifestar como ruído nas amostras, como no caso de imagens ruidosas, ou como ruído nos rótulos, como no caso de rótulos de baixa qualidade produzidos por usuários sem o conhecimento necessário para produzir as anotações.

- **Viés de quantidade (*quantity skew*)**: Decorre da possibilidade de ocorrer desbalanceamento das quantidades de dados entre os clientes. No contexto veicular, um usuário que dirige diariamente teria mais oportunidades de capturar dados do que um usuário que dirige uma vez por semana.

A consequência do viés introduzido pelo caráter não-IID dos dados utilizados no treinamento é uma degradação do desempenho do modelo treinado. Devido ao problema de heterogeneidade estatística descrito nesta subseção, não há garantia de alinhamento entre os gradientes dos modelos treinados localmente, o que pode gerar inconsistências no processo de agregação desses modelos. Isso ocorre porque em abordagens tradicionais de agregação, como o caso do FedAvg, os dispositivos clientes tipicamente realizam múltiplas etapas de treinamento, ajustando os parâmetros dos seus modelos na direção dos seus pontos ótimos locais, que são condicionados pelos seus dados não-IID. Se cada cliente direciona o ajuste do seu modelo para ótimos locais divergentes, o modelo agregado passa a se distanciar do ponto de ótimo global. Nesse caso, diz-se que ocorreu o fenômeno de desvio dos clientes (*client shift*), resultando em convergência mais lenta, instabilidade e desempenho possivelmente inferior ao dos modelos locais não agregados [Karimireddy et al., 2020, Ye e ohters, 2023].

2.4.3. Características da comunicação

As técnicas de rádio-frequência (RF) das redes atuais possuem limitações conhecidas em termos de atenuação/alcance e compartilhamento eficiente de recursos no canal, gerando alguns desafios na adoção do FL veicular. Para enfrentar estes desafios, alguns trabalhos de FL propõem modificações substanciais na infraestrutura das redes *6G & Beyond*. O artigo de Han *et al.* utiliza a proposta de redes integradas *ground-air-space* e comunicação satelital D2D (*Direct-to-Device*) para que dispositivos móveis façam o *offloading* do treinamento para satélites de órbita baixa (*Low Earth Orbit* – LEO) [Han et al., 2024]. Para evitar o curto tempo de cobertura, o sistema utiliza enlaces inter-satélites (*Inter-Satellite Links* – ISLs) para repassar os dados e modelo para os satélites vizinhos, que darão continuidade ao processo de FL naquela região.

Um desafio relacionado às técnicas de modulação e múltiplo acesso é a necessidade de suportar um número crescente de usuários em um cenário no qual o espectro é um recurso escasso. Chen *et al.* resolvem um problema de alocação de potência de *uplink* em um sistema de FL veicular na borda utilizando NOMA (*Non-Orthogonal Multiple Access*). Diferente do OFDMA, usado no 4G e 5G, um receptor NOMA cancela a interferência entre usuários que utilizam o mesmo bloco de recurso. Assim, múltiplos clientes enviam os parâmetros de um modelo ao mesmo tempo e na mesma frequência [Chen et al., 2024].

Além do desafio anteriormente citado, envolvendo o projeto de transmissores e re-

ceptores, existem desafios intrínsecos ao canal de comunicação sem fio. Em altas frequências, o esvanecimento (*fadding*) se torna mais pronunciado, aumentando a chance de um cliente perder o *deadline* de envio do modelo ou funcionar como um *straggler* que atrasa a finalização da rodada. Zejun *et al.* propõe o uso de superfícies inteligentes reconfiguráveis (*Reconfigurable Intelligent Surfaces* – RIS) para criar um linha de visada (*Line-of-Sight* – LoS) virtual entre um veículo e a ERB [Zejun et al., 2024]. Através da solução de um problema de otimização, é possível configurar dinamicamente o ângulo de reflexão de um feixe incidente em uma RIS, fazendo com que ela atue como um repetidor de sinal passivo muito mais energeticamente eficiente do que alternativas ativas, como o uso de *drones* (*Unmanned Aerial Vehicles* – UAVs). Os autores argumentam que a técnica aumenta a quantidade de dados utilizados no treinamento, o que é essencial para tornar o modelo mais útil e representativo.

Existem alguns desafios em aberto quanto aos protocolos utilizados para FL veicular. A camada de aplicação precisa lidar com clientes e servidores que se desconectam ou demoram muito tempo para enviar suas contribuições. Zhangchen *et al.* propõem um protocolo para FL que modifica toda a FSM de troca de mensagens em um cenário P2P [Xu et al., 2024b]. Utilizando privacidade teórica da informação (*Information-theoretic privacy*), os autores provam a tolerância à falhas bizantinas na presença de f nós maliciosos se o número total de nós for $N = 3f + 2$.

Alguns trabalhos demonstram limitações de desempenho importantes do TCP em condições de redes desafiadoras. O atraso introduzido pelo *three-way handshake* é inadequado quando o contato entre cliente e servidor ocorre em um período de tempo muito breve, o que é comum em locais com cobertura incompleta. Valores baixos de janela de congestionamento em redes sem fio também foram reportados em experimentos da literatura [Thomaz et al., 2024]. Um trabalho recente de Chen *et al.* propôs um protocolo de transporte disruptivo, com controle de congestionamento otimizado especificamente para aprendizado distribuído, com foco em envio de parâmetros 1:N (*incast*) [Chen et al., 2023b].

Por fim, as redes IP não proveem garantias como cache, agregação de pacotes, *multicast* nativo, endereçamento móvel e Notificação Explícita de Congestionamento (*Explicit Congestion Notification* – ECN). Amadeo *et al.*, por exemplo, mostram que as Redes Centradas em Informação (*Information Centric Networks* – ICNs) permitem com que a própria rede participe ativamente da descoberta, seleção e inicialização dos clientes do FL [Amadeo et al., 2025]. A arquitetura *Named Data Entity* (NDN) define um esquema próprio de controle de congestionamento na camada de transporte, chamado de PCON (*Practical Congestion Control*), que implementa Gerenciamento Ativo de Fila (*Active Queue Management* – AQM) nas interfaces dos roteadores da ICN NDN para permitir ECN. Um dos diferenciais do artigo é o fato de que os resultados comparativos entre mecanismos de controle de congestionamento (TCP BIC, TCP CUBIC, TCP Westwood e PCON) demonstram que a topologia de rede influencia o desempenho, sugerindo que o projeto adequado da interconexão entre elementos no núcleo também é um desafio.

2.4.4. Características da aplicação

O paradigma do FL difere do CL não somente pela distribuição do treinamento dos modelos, mas também pelo processo de coleta e anotação dos dados utilizados. No CL, a ausência de restrições sobre a transmissão de dados permite que o custo associado à rotulação seja abstraído do usuário final do sistema, uma vez que torna possível o envio desses dados para que terceiros produzam as respectivas anotações. Já no caso do FL, a premissa de privacidade que justifica a abordagem impede a transferência dessa responsabilidade, centralizando no usuário todo custo envolvido no procedimento de rotulação.

O problema de falta de rótulos é particularmente acentuado no contexto veicular. Muitas tarefas de interesse como detecção de veículos, rastreamento de objetos e segmentação semântica dependem de rótulos complexos, envolvendo a definição de caixas delimitadoras para cada objeto na cena capturada ou a produção de máscaras de segmentação, atribuindo uma classe a cada instância em ambos os casos. Esse elevado custo associado à produção de anotações promove um cenário de escassez de rótulos. No entanto, essa escassez não necessariamente se distribui de forma homogênea pelo sistema. É possível, por exemplo, que o servidor tenha acesso a conjuntos de dados públicos que contenham informação relevante ou mesmo dados que tenham sido previamente coletados e rotulados, servindo como uma possível referência para o treinamento. Apesar de os dados coletados pelos clientes serem majoritariamente não rotulados, alguns clientes podem dispor de recursos para realizar a rotulação de uma parcela ou a totalidade dos seus dados.

Trabalhos na literatura mapeiam, exploram e propõem soluções para diferentes cenários de rotulação, sendo os casos de rótulos nos clientes ou rótulos no servidor os mais predominantes [Jeong et al., 2021, Ji et al., 2024]. A Figura 2.5 apresenta uma taxonomia das diferentes combinações de dados rotulados e não rotulados pertencentes aos participantes no treinamento federado, relacionando os cenários com abordagens de aprendizado federado semissupervisionado (*Federated Semi-Supervised Learning* – FSSL).

Outro problema que dificulta o treinamento dos modelos utilizados em contextos veiculares, especialmente quando voltados para tarefas de visão computacional, é que comumente essas aplicações impõem restrições de latência severas, devendo ser capazes de operar em tempo real. Um modelo de detecção de objetos que contribui para a tomada de decisões de um sistema de direção autônoma deve ser capaz de processar o fluxo de imagens recebido da câmera em tempo suficientemente curto para que não atrase o restante do sistema. Para atender a esses requisitos, detectores de objetos de um estágio são frequentemente escolhidos, uma vez que comumente apresentam menor tempo de inferência quando comparados aos detectores de dois estágios.

Entretanto, detectores de um estágio apresentam limitações quando utilizados no contexto de aprendizado semissupervisionado (*Semi-Supervised Learning* – SSL). Técnicas propostas para esse contexto tipicamente dependem da geração de pseudo-rótulos, ou seja, utilizam rótulos gerados pelo próprio modelo em seu treinamento. Em detectores de um estágio, a ausência de uma etapa separada de geração de regiões de interesse e de refinamento dessas predições torna a filtragem de pseudo-rótulos de melhor qualidade mais difícil, potencialmente levando ao uso de anotações incorretas ou desbalanceadas [Zhang et al., 2022d, Kim et al., 2023]. Dessa forma, a construção de estratégias de

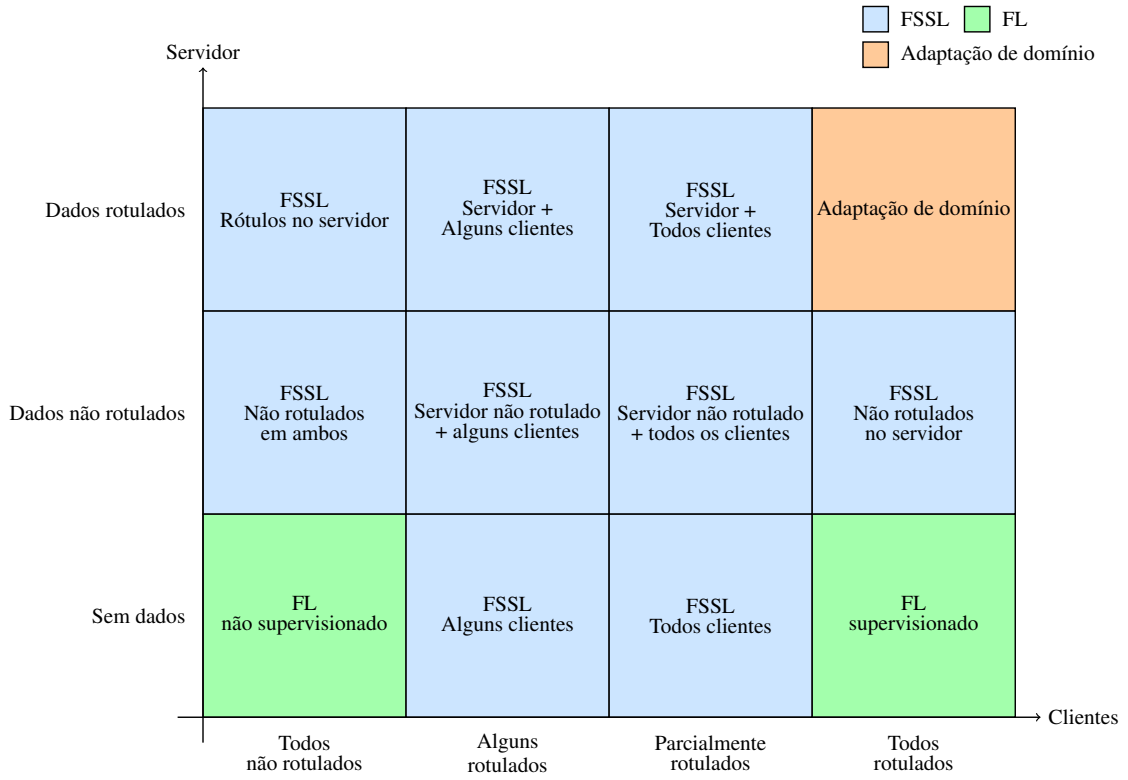


Figura 2.5. Taxonomia de cenários de aprendizado federado em diferentes cenários de disponibilidades de rótulos. Blocos azuis representam casos típicos de aplicações de FSSL. Blocos verdes representam casos de FL clássico. Já o bloco laranja representa um caso específico de aplicação da técnica de adaptação de domínio.

FSSL que suportem arquiteturas capazes de atender aos requisitos de latência das aplicações veiculares e sejam capazes de lidar com a baixa qualidade dos rótulos gerados por essas arquiteturas é um desafio adicional para o desenvolvimento de sistemas de VFL.

2.5. Atividade Prática

Esta seção descreve a atividade prática do minicurso, que foi concebida como uma simulação realista e detalhada do aprendizado federado aplicado especificamente ao contexto veicular, integrando de forma coesa os aspectos teóricos discutidos anteriormente. Primeiramente, apresenta-se o conjunto de dados para a tarefa de aprendizado que será abordada na atividade prática. O treinamento prático inicia-se com a demonstração de como simular a heterogeneidade dos clientes, abordando tanto as diferenças de hardware entre os clientes quanto a diversidade nos volumes e tipos de dados coletados por cada veículo. Esta etapa é crucial para que os participantes compreendam como os modelos de inteligência artificial se comportam diante da variabilidade encontrada em redes veiculares de larga escala no mundo real. Para conferir autenticidade ao experimento, a atividade aborda a geração de traços de mobilidade por meio do *Simulation of Urban MObility* (SUMO)[1] [Lopez et al., 2018], uma ferramenta para simular o movimento de veículos em ambientes urbanos. Em conjunto com a aplicação de modelos de canal de

[1]Disponível em: https://eclipse.dev/sumo/

comunicação, esses traços serão utilizados para gerar estatísticas precisas sobre os clientes, oferecendo uma base de dados sólida para a análise do sistema. Além disso, o curso utiliza ferramentas de visualização, que permitem a demonstração clara e intuitiva dos fluxos de mobilidade e das dinâmicas de comunicação que ocorrem durante o processo de aprendizado.

A fase final da parte prática foca na implementação técnica utilizando o arcabouço Flower [Beutel et al., 2020], adaptado para incorporar os traços de computação, mobilidade e comunicação gerados nas etapas anteriores. Por meio desta plataforma, os participantes têm a oportunidade de avaliar o impacto real dos diferentes parâmetros configurados no aprendizado federado, observando como cada ajuste influencia o desempenho final do modelo treinado de forma distribuída. Essa abordagem permite uma compreensão profunda das técnicas de otimização necessárias para garantir que os sistemas inteligentes de transporte operem com máxima eficiência e segurança. Esta seção descreve as etapas a serem executadas ao longo da atividade prática. A definição dos pré-requisitos, dados necessários e comandos a serem executados estão disponíveis no repositório desse capítulo[2].

Ao fim da atividade prática, os participantes compreenderão de forma abrangente e aplicada o funcionamento das ferramentas essenciais apresentadas, estando plenamente capacitados para adaptá-las à implementação de novas e avançadas técnicas de aprendizado federado no contexto veicular.

2.5.1. Definição do cenário federado

O conjunto de dados a ser utilizado na atividade prática será o *Traffic Signs Preprocessed*[3] [Stallkamp et al., 2011]. Este conjunto de dados possui próximo a 50.000 imagens de treino e 12.000 imagens de teste e tem como tarefa de aprendizado a classificação de placas de trânsito alemãs, divididas em 43 classes. Assim, o classificador recebe uma imagem de 32 x 32 pixeis com 3 canais de cores RGB e retorna a classe mais provável. Para o cenário abordado na prática, utilizam-se 20 clientes, cujas definições de dados e dispositivos são apresentadas a seguir.

2.5.2. Modelagem da heterogeneidade de clientes

No ambiente federado, há duas fontes principais de heterogeneidade: os dados coletados e os dispositivos. Como visto na Seção 2.4, a heterogeneidade dos dados está relacionada ao ambiente em que os dados foram coletados, ao método de coleta, às configurações e ao equipamento utilizado. Por outro lado, a heterogeneidade de dispositivos está relacionada a características físicas, como a frequência de processamento, a quantidade de memória e o número de núcleos de processamento disponíveis.

2.5.2.1. Heterogeneidade de Dados

Ao contrário de cenários centralizados nos quais um servidor armazena o conjunto de dados completo, no aprendizado federado considera-se que os dados estão distribuídos

[2]Disponível em: https://github.com/GTA-UFRJ/MinicursoAprendizadoFederadoVeicularSBRC2026

[3]Disponível em: https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-preprocessed

entre os dispositivos participantes do treinamento. Em tarefas de classificação de imagens, como a abordada na atividade prática, os clientes possuem conjuntos de dados sem interseção, uma vez que a captura dos dados depende de sua posição, condições da câmera e do ambiente no momento da captura.

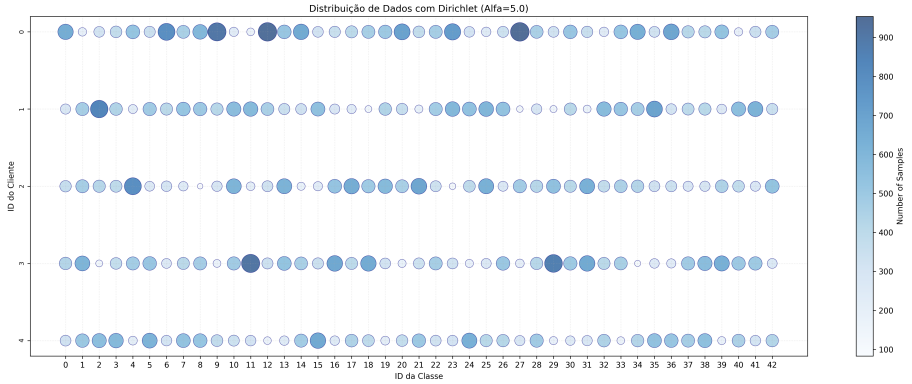


Figura 2.6. Distribuição dos dados de clientes ao utilizar a função de Dirichlet com $\alpha = 5$. O conjunto de dados gerados é mais homogêneo quando o valor do parâmetro α é maior.

Assim, a distribuição de dados afeta o desempenho de propostas de aprendizado federado. Cenários realistas de treinamento consideram a variação da distribuição de dados, que é inexistente em conjunto de dados capturados para realizar o treinamento centralizado. Portanto, para gerar subconjuntos sem interseção e simular a heterogeneidade de dados, a função de Dirichlet [Ferguson, 1973] é amplamente utilizada. A função recebe dois parâmetros, o número de conjuntos de dados esperado e o $0 \leq \alpha < \infty$. O número de conjuntos de dados será igual ao número de clientes presentes na simulação, enquanto o α é o parâmetro responsável por controlar o grau de heterogeneidade dos dados. Um valor alto de α, tipicamente maior do que 5, gera distribuições semelhantes, como exibido na Figura 2.6, enquanto valores menores geram conjuntos de dados mais desbalanceados, como exibido na Figura 2.7.

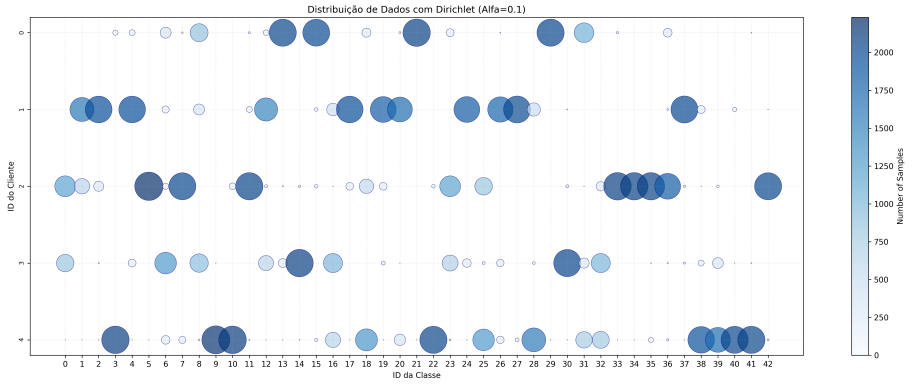


Figura 2.7. Distribuição dos dados de clientes ao utilizar a função de Dirichlet com $\alpha = 0,1$. O conjunto de dados gerados é desbalanceado segundo o número de amostras de cada classe para diferentes clientes.

Dessa forma, pode-se gerar conjuntos de dados heterogêneos, com dados únicos,

para executar o treinamento simulando a existência de diversos clientes. Além disso, a heterogeneidade de dados abrange um escopo maior do que o cenário veicular, aplicada em diversos cenários de aprendizado federado estáticos ou dinâmicos. A seguir, discute-se como simular a heterogeneidade dos dispositivos dos clientes.

2.5.2.2. Heterogeneidade de Dispositivo

Em um cenário veicular, como visto na Seção 2.4.1, os clientes apresentam diferentes dispositivos. Veículos autônomos de níveis baixos realizam algumas unidades de teraoperações por segundo (TOPS), enquanto que veículos completamente autônomos executam milhares de teraoperações. Dessa forma, veículos que possuem maior capacidade computacional, ao treinarem o mesmo modelo de aprendizado profundo, atualizam de forma mais rápida do que veículos com menor capacidade computacional.

Assim, nesta parte da atividade prática modela-se por meio de um modelo matemático o tempo de processamento de clientes no aprendizado federado veicular a partir das especificações do dispositivo do cliente. O modelo matemático permite que o sistema seja executado em uma mesma máquina, o que é necessário para casos em que não há disponibilidade de dispositivos heterogêneos. Caso haja disponibilidade de diversos dispositivos, é possível substituir o modelo matemático pelo tempo de computação do próprio dispositivo. Outra parte essencial para o VFL é a simulação da mobilidade dos clientes, para a qual utiliza-se um simulador de mobilidade discutido a seguir.

2.5.3. Simulação da mobilidade de veículos com o SUMO

Uma vez definidas as características de dispositivo, deve-se simular o movimento dos veículos de forma realista. O simulador SUMO (*Simulation of Urban MObility*)[4] modela características microscópicas da dinâmica de tráfego urbano. O SUMO calcula o comportamento de cada veículo de forma individual e, com isso, também considera a interação entre eles. Elementos como interseções, sinais, placas de trânsito, pedestres podem ser adicionados para adicionar mais elementos que corroboram uma simulação mais realista. Uma simulação é construída com base na escolha do mapa e na definição do comportamento dos veículos na simulação. A simulação retorna uma lista com a posição de todos os veículos ao longo da simulação, e outras ferramentas utilizam essa lista para determinar a posição dos elementos no instante em que se realiza uma transmissão.

O site do SUMO possui uma documentação bem detalhada sobre os diversos elementos que podem ser adicionados na simulação para tornar os dados mais próximos de reais. Os principais parâmetros necessários para a configuração da simulação são o movimento dos veículos, quantidade de veículos na simulação, velocidade máxima das vias e o mapa a ser utilizado. Além disso, é possível configurar pontos de interesse, como estações rádio base e pontos de parada.

Para a geração dos movimentos dos veículos utiliza-se a ferramenta **randomTrips**. A partir da especificação da quantidade de veículos e informações sobre a velocidade e taxa de entrada, a ferramenta gera voltas aleatórias no mapa especificado. A

[4]https://eclipse.dev/sumo/

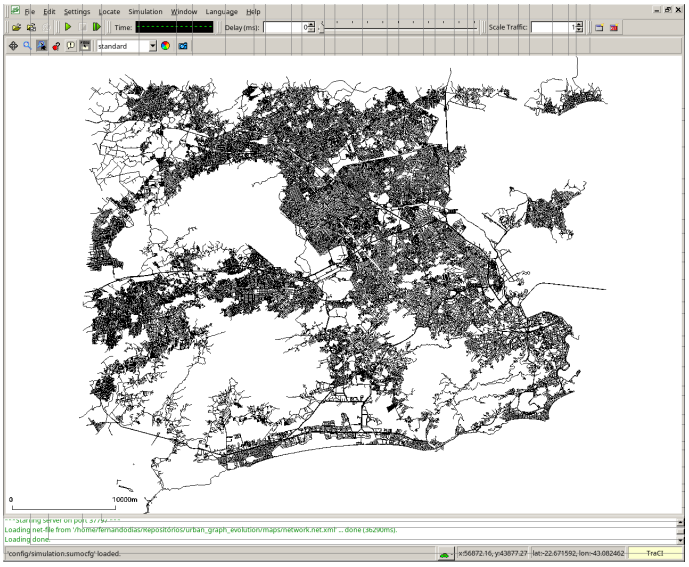


Figura 2.8. Ilustração do uso do mapa da região metropolitana do Rio de Janeiro.

fim de manter todos os clientes ao longo do treinamento, configura-se a ferramenta para que os veículos entrem na simulação ao mesmo tempo. Além disso, para simplificar a execução, todos os veículos viajam com a velocidade máxima da via.

O mapa é um elemento que pode ser obtido de um cenário real ou criado sinteticamente. Por exemplo, o mapa exibido na Figura 2.8 exibe o mapa real da cidade do Rio de Janeiro. Esse mapa é obtido por meio do OSM (*Open Street Map*)[5], uma ferramenta para exportar mapas realistas com informações precisas sobre velocidade de vias e pontos de interesse. Entretanto, para fins práticos é comum utilizar cenários sintéticos, que simulam a cidade de Manhattan, por exemplo. Assim, nessa atividade prática, utiliza-se um mapa quadrado de 1000 x 1000 m^2, configurado de forma que as rotas sejam circulares, para evitar a saída dos clientes ao longo da simulação do aprendizado federado, com uma estação rádio base na origem como um ponto de interesse.

A seguir, utilizam-se os dados gerados como entrada do modelo de comunicação para determinar a vazão dos clientes e verificar o impacto no treinamento federado veicular.

2.5.4. Simulação da comunicação

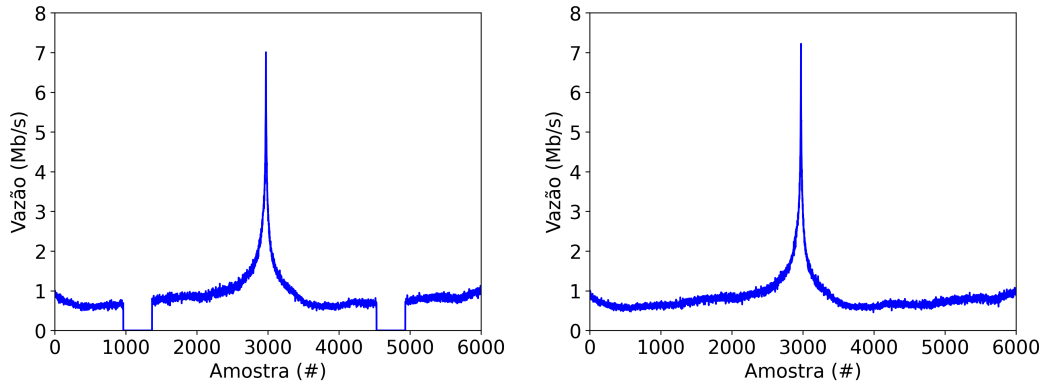
O modelo de comunicação recebe a saída processada do SUMO para determinar os atrasos de comunicação dos clientes. Para o modelo de comunicação, utiliza-se o 3GPP TR 38.901 [Zhu et al., 2021a, Chatzoulis et al., 2023], o modelo de canal mais versátil e amplamente aceito. Ele é capaz de lidar com cenários urbanos, rurais e rodoviários com parâmetros realistas para desvio Doppler, perda de percurso (*path loss*) e propagação multicaminho.

A implementação para modelar o canal 3GPP TR 38.901 para o código de comunicação sem fio foca nos modelos de perda de percurso (*path loss*) e desvanecimento

[5]https://www.openstreetmap.org

(*fading*) de Macrocélula Urbana (UMa - Urban Macrocell) e Microcélula Urbana (UMi - Urban Microcell). Assim, a implementação inclui o cálculo de perda de percurso (*path loss*) para cenários com visada direta (*Line-of-Sight* – LOS) e sem visada direta (*Non-Line-of-Sight* – NLOS), desvio Doppler (Doppler *shift*) para consideração da mobilidade, desvanecimento de pequena escala (*small-scale fading*) utilizando os modelos de desvanecimento de Rician e Rayleigh e sombreamento (*shadowing*) para desvanecimento de larga escala (*large-scale fading*).

Para a perda de percurso, considera-se o modelo de Macrocélula Urbana (UMa - Urban Macrocell) do 3GPP TR 38.901, que inclui condições LOS e NLOS com efeitos de sombreamento. Para o desvio Doppler, precisa-se simular o efeito do movimento relativo entre o usuário e a estação rádio-base. Esse movimento é calculado com base na velocidade do cliente e frequência da portadora. Assume-se o desvanecimento de pequena escala considerando o desvanecimento Rician e o desvanecimento Rayleigh. Para detalhes sobre o modelo matemático utilizado, consulte o repositório de referência[6].



(a) Vazão em um cenário com desconexão de 11 segundos.

(b) Vazão em um cenário com conexão constante.

Figura 2.9. Visualização da vazão média de um cliente executando a ferramenta de simulação com uma estação rádio base.

Portanto, utiliza-se o modelo de mobilidade como a entrada do modelo de comunicação para gerar os atrasos relativos ao sistema, ilustrados na Figura 2.9. Na Figura 2.9(a)

A seguir, discute-se como adaptar o arcabouço Flower para incluir os atrasos de computação e comunicação gerados para simular a heterogeneidade e mobilidade dos clientes.

2.5.5. Adaptação do arcabouço Flower para o treinamento federado veicular

A simulação com o arcabouço se baseia no desenvolvimento de dois arquivos principais, um para o cliente e um segundo para o servidor. No lado do cliente, é necessária a implementação de ao menos dois métodos, o método para o ajuste dos pesos do modelo e o método de avaliação do modelo. O primeiro método é executado no início de cada rodada, sendo invocado pelo servidor por meio de uma chamada gRPC. O segundo método é invocado no fim de cada rodada para testar o desempenho do modelo treinado até

[6]Disponível em https://github.com/AiramL/TimeOptimizedFederatedLearning

o momento. Ambas as funções permitem o envio de dicionários de parâmetros, os quais podem ser utilizados para a implementação de estratégias no servidor.

O servidor, por sua vez, utiliza um objeto que deve implementar a estratégia do aprendizado federado. Assim, a estratégia define como realizar a agregação dos modelos de diferentes clientes, selecionar os clientes e processar parâmetros enviados. Para isso, a estratégia possui métodos que implementam a fase de treino e de avaliação de clientes. Em cada fase, existe um método de seleção de clientes que pode variar, por exemplo, na quantidade de clientes selecionados em cada fase.

Portanto, a adaptação do arcabouço para o aprendizado veicular consiste em modificar os métodos relativos ao cliente para incluir parâmetros para simular a mobilidade e a heterogeneidade dos dispositivos. Assim, os clientes devem simular o atraso do recebimento do modelo, a atualização local dos parâmetros e o reenvio ao servidor. Ao variar os parâmetros de mobilidade, comunicação e computação, é possível verificar o impacto na eficiência e tempo de treinamento ao utilizar diferentes estratégias adaptadas ao VFL.

2.6. Tendências de Pesquisa

Esta seção apresenta as principais tendências de pesquisa em aprendizado federado para solucionar os desafios práticos abordados na Seção 2.4. As tendências são divididas em seis linhas de pesquisa e analisadas conforme a viabilidade de implementação em um cenário veicular. Inicialmente, abordam-se as propostas de seleção de clientes que substituem a seleção aleatória. Após isso, discutem-se estratégias que alteram a função de agregação de modelos para consideração da heterogeneidade de dados. Além disso, são discutidos a personalização de modelos e a sua subárea de pesquisa, o agrupamento de clientes para o aumento da acurácia do modelo e redução do tempo de convergência quando os clientes possuem distribuições de dados heterogêneos. Ainda, apresenta-se a alternativa para o treinamento federado com múltiplos modelos, a fim de mitigar problemas ocasionados pela heterogeneidade dos dispositivos ou do canal de comunicação de clientes. Por fim, são apresentados os novos paradigmas de aprendizado que buscam solucionar problemas específicos de aplicações, por exemplo, lidar com rótulos faltantes.

2.6.1. Seleção de clientes

A seleção de clientes no aprendizado federado consiste em definir um subconjunto de clientes para participar de uma determinada rodada de treino. É comum, especialmente em cenários veiculares, a heterogeneidade nas características dos clientes. Elementos como a distribuição de dados locais, a capacidade computacional dos dispositivos de treino de cada cliente, a qualidade da conexão e até mesmo a distribuição geográfica dos clientes são fatores que impactam a qualidade dos gradientes locais, afetando o modelo global e o tempo de treino de uma rodada. Os métodos de seleção de clientes são baseados em: otimização de recursos, importância estatística , aprendizado por reforço e reputação [Soltani et al., 2022, Fu et al., 2023, Li et al., 2024]. Os métodos baseados em reputação estão fora do escopo deste minicurso, pois visam um cenário no qual os clientes podem ser maliciosos.

2.6.1.1. Métodos baseados em otimização de recursos

Os métodos baseados em otimização de recursos formulam uma função objetivo a ser maximizada ou minimizada a fim de selecionar os melhores clientes. Esses métodos podem considerar diversas métricas na formulação do problema de otimização, como a capacidade computacional, largura de banda disponível ou localização geográfica dos clientes. Nem sempre a solução ótima dos problemas propostos pode ser calculada em tempo hábil, sendo comuns aproximações da solução por algoritmos gulosos.

Nishio e Yonetani propõem um protocolo para seleção de clientes no aprendizado federado [Nishio e Yonetani, 2019] no qual os clientes com maiores capacidades de processamento e menor latência de comunicação são priorizados no esquema de seleção apresentado. [Buyukates e Ulukus, 2021] consideram um cenário de aprendizado federado no qual o servidor de agregação realiza uma seleção aleatória e descarta parte dos modelos treinados pelos clientes. O objetivo é aumentar a velocidade de treinamento do aprendizado federado selecionando os primeiros *m* clientes que enviam o modelo ao servidor de agregação. A vantagem da proposta é utilizar uma heurística simples que dispensa a formulação de um problema de otimização para a seleção de clientes. No entanto, a proposta introduz um desperdício computacional para os clientes retardatários que foram selecionados.

[Su et al., 2024] propõem um algoritmo de seleção de clientes para o *Online Federated Learning* (OFL), chamado LCCS (*Low-Cost Client Selection*). Os autores formulam um problema de otimização para maximizar a utilidade do modelo e minimizar o custo de comunicação. A proposta é relevante para cenários veiculares, pois as condições de trânsito, como o engarragamento, geram áreas com mais clientes. Logo é interessante otimizar a largura de banda para reduzir o custo de comunicação.

O TOFL (*Time Optimized Federated Learning*) [de Souza et al., 2025] é uma proposta para seleção de clientes no aprendizado federado veicular que considera a mobilidade dos clientes. Como a posição dos clientes é dinâmica, suas condições de conexão variam. Isso aumenta o tempo de treinamento do modelo devido ao atraso de comunicação com o servidor. Assim, os autores formulam um problema de otimização que considera os atrasos de cada cliente para selecionar os melhores em uma rodada a fim de reduzir o tempo total de treinamento em cada rodada. No entanto, o atraso é uma variável desconhecida, pois é obtido apenas após a seleção. Portanto, o TOFL, a partir de informações sobre as vazões anteriores dos clientes, utiliza um modelo LSTM para prever a vazão futura dos clientes e determinar os próximos atrasos. A seleção, apesar de reduzir o tempo de treinamento em determinadas condições, possui limitações, como a qualidade do estimador em determinar as condições dos clientes. Uma alternativa para evitar o descarte de atualizações de forma mais eficiente é executar o estimador de atrasos no cliente, como feito em CAIROS [de Souza et al., 2026]. Assim, os clientes podem enviar antecipadamente o modelo quando detectam que as condições de comunicação estão se degradando. Entretanto, como a proposta aumenta a complexidade do cliente, CAIROS é inadequado para o caso em que os clientes possuem recursos computacionais limitados.

Apesar dos métodos reduzirem o tempo de execução de uma rodada, a formulação proposta desconsidera o número de rodadas totais. Isso ocorre pois os métodos otimizam

a seleção de cliente considerando a heterogeneidade de sistema, isto é, a heterogeneidade dos dispositivos participantes. Por outro lado, o número total de rodadas pode ser otimizado considerando a relevância dos dados dos clientes para a atualização do modelo global. Dessa forma, há um compromisso entre o número de rodadas e o tempo de cada rodada, resolvido pelos métodos baseados em importância estatística.

2.6.1.2. Métodos baseados em importância estatística

A qualidade dos dados utilizados para o treinamento dos modelos impacta o desempenho do modelo treinado, além de reduzir a quantidade de rodadas totais para o treinamento. Assim, outros métodos de seleção de cliente buscam clientes com maior influência positiva na atualização do modelo global. O objetivo de métodos baseados em importância é acelerar a convergência do modelo global através da seleção dos clientes que possuem dados mais relevantes. Em alguns casos, esses métodos podem também considerar a otimização de recursos computacionais, como em Oort [Lai et al., 2021], que considera a heterogeneidade do sistema na formulação do problema de otimização.

[Luo et al., 2022] e [Lai et al., 2021] propõem esquemas de seleção de clientes que otimizam a velocidade de convergência do modelo em ambientes de aprendizado federado. Selecionar clientes baseando-se unicamente na representatividade dos dados diminui o número total de épocas para a convergência do modelo. No entanto, clientes que possuem dados mais relevantes para o problema podem ter um tempo de treinamento mais longo a cada época, de acordo com a quantidade de dados e as características do dispositivo utilizado. Por um lado, o aumento do tempo entre as épocas implica um atraso geral maior, uma vez que os ambientes de FL são geralmente síncronos e aguardam todos os clientes enviarem seus modelos locais ou atingirem o tempo limite. Por outro lado, selecionar clientes com maior poder computacional para reduzir o tempo entre as rodadas pode resultar em mais rodadas para a convergência, caso os clientes selecionados possuam dados estatisticamente insignificantes para a tarefa de aprendizado. Existe um compromisso entre o número de épocas para o treinamento do modelo e o tempo total de cada rodada. Assim, os autores consideram simultaneamente as características dos dispositivos e a relevância da distribuição de seus dados para reduzir o tempo de convergência do modelo global. Enquanto Luo *et al.* propõem uma função não convexa em relação ao número esperado de épocas, baseada nas atualizações anteriores dos clientes, [Lai et al., 2021] evitam a programação linear, criando uma função de seleção que avalia as perdas de treinamento dos clientes.

FedSA [Neto et al., 2022] utiliza uma meta heurística baseada no arrefecimento simulado para o ajuste dos hiperparâmetros e a seleção eficiente de clientes. A partir do estado atual do modelo, a proposta seleciona um grupo de clientes para realizar o treinamento na rodada atual. Já [Rai et al., 2022] propõem a métrica de amostragem por irrelevância para seleção de clientes, visando melhorar a acurácia final de modelos de aprendizado federado em cenários IID e não-IID. O objetivo é selecionar clientes considerando a qualidade e a quantidade de suas amostras. Cada cliente calcula sua métrica de amostragem por irrelevância e a envia ao servidor. O servidor, então, agrupa os clientes de acordo com o valor informado em três clusters: positivo, negativo e zero. Finalmente,

a metodologia proposta seleciona aleatoriamente os clientes de cada cluster para alcançar uma convergência mais rápida do modelo.

Porém, os métodos baseados em importância estatística são muito complexos devido à estocacidade e à não causalidade do problema, uma vez que o servidor obtém a acurácia de teste do modelo após a seleção. Assim, uma forma de reduzir a complexidade do problema da seleção de clientes é delegar a tarefa a um modelo de aprendizado por reforço.

2.6.1.3. Métodos baseados em aprendizado por reforço

Métodos baseados em aprendizado por reforço profundo (*Deep Reinforcement Learning – DRL*) utilizam redes neurais profundas para abstrair políticas de seleção de clientes. Esses métodos formulam um espaço de estados e de ações onde o modelo de DLR é treinado, aprendendo padrões intrínsecos do cenário em uma representação de alta dimensionalidade a partir da interação com o ambiente.

FAVOR [Wang et al., 2020a] utiliza DRL na seleção de clientes em um cenário onde os clientes possuem uma distribuição de dados locais não IID. A abordagem consiste em treinar uma rede (*Deep Q-Network – DQN*) em um cenário onde o espaço de estados é definido como o conjunto de pesos do modelo global e o conjunto de pesos do modelo local de cada cliente. Devido à alta dimensionalidade resultante, aplica-se uma técnica de redução de dimensionalidade sobre esses pesos antes que o agente selecione o subconjunto ideal para o treinamento. Já os autores em [Zhang et al., 2021] utilizam o algoritmo *Deep Deterministic Policy Gradient* para realizar DRL para a seleção de clientes em um cenário de (*Internet of Things* – IoT) considerando o custo de treinamento, de comunicação e a qualidade dos modelos locais.

Um cenário veicular é abordado em [Moon e Lim, 2024], onde clientes realizando computação veicular (*Vehicular Edge Computing – VEC*) treinam um modelo em aprendizado federado que será enviado a um servidor para agregação. A função de recompensa considera a acurácia do modelo global no *dataset* de teste, enquanto a observação considera o tempo de processamento total e a sobrecarga de comunicação em uma rodada para modelar um cenário de aprendizado por reforço. O algoritmo DQN foi utilizado para treinar um modelo de DRL nesse cenário. Ademais, o trabalho apresenta outro modelo de seleção baseado em lógica *fuzzy*.

A seleção de clientes é desafiadora devido à sua complexidade, principalmente em ambientes veiculares, onde o atraso por rodada é dependente da mobilidade dos clientes selecionados. Além disso, reduzir a complexidade da aplicação do cliente é necessário em casos de computação limitada. Assim, uma forma de reduzir o tempo de convergência do modelo, mantendo a aplicação do cliente simples, é modificar no servidor a função utilizada para a agregação, considerando a heterogeneidade dos clientes.

2.6.2. Estratégias de modificação da função de agregação

O servidor de agregação no FedAvg realiza uma média ponderada dos pesos da rede neural para a agregação de diferentes modelos. A contribuição de cada cliente na

média ponderada é dada pela quantidade de dados que o cliente possui sobre a quantidade de dados totais utilizados para a agregação da rodada atual. Entretanto, essa estratégia não é ótima quando os clientes possuem dados ou dispositivos heterogêneos. Assim, novas estratégias buscam otimizar a contribuição dos clientes na agregação a fim de reduzir o tempo de convergência e aumentar o desempenho do modelo.

Uma das primeiras propostas de agregação para substituir o FedAvg foi o FedProx [Li et al., 2020]. A proposta modifica a função de agregação para lidar com a heterogeneidade de dispositivos e de dados. Para evitar que as atualizações dos clientes retardatários sejam descartadas, os autores propõem o uso de épocas locais variáveis, ajustadas à capacidade computacional do cliente. Além disso, o desafio da heterogeneidade de dados é resolvido por meio da aplicação de um termo proximal para regularizar as atualizações. Esse termo é otimizado e aplicado localmente pelo cliente para considerar o impacto da heterogeneidade de dados e da quantidade distinta de épocas locais executadas na atualização. O objetivo do termo proximal, que regulariza o modelo, é produzir atualizações que sejam próximas do modelo global da rodada, garantindo maior consistência.

A proposta do SCAFFOLD (*Stochastic Controlled Averaging algorithm*) [Karimireddy et al., 2020] também modifica como os modelos são agregados ao aplicar uma variável de regularização para reduzir a variância das atualizações. A cada passo local, a variável de controle é aplicada para evitar que o modelo local se distancie do modelo global. A variável de controle é armazenada por cada cliente e pelo servidor, a fim de compartilhar o estado entre rodadas diferentes, sendo atualizada ao final de cada rodada. Dessa forma, a variável de controle possui um efeito similar ao do termo proximal do FedProx. No entanto, como a variável de controle atua no espaço de gradientes, esta permite uma maior exploração do espaço de soluções. Por outro lado, o termo proximal atua no espaço do modelo e reduz o tamanho dos passos, aumentando o tempo de convergência em comparação com o SCAFFOLD.

FedNova (*Federated Normalized averaging*) [Wang et al., 2020b] é uma proposta que assume que os clientes podem executar um número diferente de épocas locais de acordo com as condições do dispositivo. Assim, para resolver a inconsistência causada pela diferença na quantidade de atualizações locais, os autores propõem uma normalização da atualização, considerando a quantidade de computação realizada por cada cliente. Essa estratégia é executada diretamente no servidor, diferentemente do FedProx e do SCAFFOLD que modificam a arquitetura do cliente diretamente.

MOON (*MOdel-cOntrastive federated learNing*) [Li et al., 2021c] modifica a função de perda dos clientes ao utilizar três modelos para treinar: o modelo global, o modelo local e o modelo local da rodada anterior. Assim, os clientes aplicam o aprendizado constrativo, no qual a função de perda busca aproximar as representações geradas pelo modelo local atual às representações do modelo global, além de afastá-las das representações do modelo local da rodada anterior. Por fim, a perda constrativa é somada à perda supervisionada para atualizar os pesos do modelo.

No FedFTG (*Federated Fine-tuning Global model*) [Zhang et al., 2022c], antes de enviar o modelo agregado aos clientes, o algoritmo aplica a destilação de conhecimento usando os modelos dos clientes como professores do modelo global. O objetivo é au-

mentar o desempenho do modelo agregado, principalmente quando os clientes possuem distribuições de dados heterogêneas.

Entretanto, as propostas de novas estratégias de agregação assumem o aprendizado de um único modelo global, o que pode ser prejudicial para o sistema quando os clientes possuem dados e dispositivos altamente heterogêneos. Assim, a personalização de modelos é mais adequada, pois permite mitigar os problemas de heterogeneidade no modelo global ao mesmo tempo que produz um modelo local mais adequado aos dados e dispositivos dos clientes.

2.6.3. Personalização de modelos de inteligência artificial

O aprendizado federado tradicional, baseado no FedAvg [McMahan et al., 2017], otimiza um modelo global único que maximize o desempenho médio entre todos os participantes. No entanto, em redes veiculares, a premissa de que um único modelo é capaz de atender a todos os clientes é frequentemente invalidada pela heterogeneidade dos dados [Li et al., 2021a]. Veículos operando em diferentes regiões geográficas, sob condições climáticas variadas ou com perfis de condução distintos, geram distribuições estatísticas distintas. Nesse contexto, o Aprendizado Federado Personalizado (Personalized Federated Learning, PFL) surge como um paradigma que visa encontrar um equilíbrio ótimo entre o conhecimento global compartilhado e a especialização local necessária para cada veículo.

O FedDLD (*Federated Dual-Level Distillation*) [Xiao et al., 2025] é uma proposta que combina o aprendizado federado veicular com a destilação de conhecimento para reduzir os problemas causados pela heterogeneidade dos dados. Na proposta, os clientes recebem o modelo global e realizam a destilação localmente para personalizar o modelo de acordo com os dados locais. Além disso, os clientes são agrupados hierarquicamente de acordo com a similaridade dos modelos gerados no treinamento local. Essa etapa tem como objetivo realizar uma destilação mútua e adaptativa entre os clientes que possuem distribuições de dados similares. O servidor, por sua vez, os modelos locais e realiza a agregação. Para garantir a segurança do sistema, o FedDLD considera a reputação dos clientes durante a agregação e armazena as versões dos modelos globais em uma corrente de blocos em forma de grafo direcionado acíclico.

Embora promissores, métodos convencionais de personalização restringem-se a ajustes finos locais ou à aplicação de termos de regularização que tentam equilibrar a divergência entre os pesos locais e o modelo global. No entanto, no cenário veicular, tais métodos enfrentam o desafio crítico da divergência de pesos (*weight drift*), onde a otimização para distribuições locais heterogêneas pode prejudicar a estabilidade da agregação entre os nós. Nesse contexto, linhas de pesquisa recentes avançam em direção a promoverem adaptações arquiteturais e semânticas profundas, capazes de lidar de forma mais robusta com a complexidade dos dados e as restrições de rede.

2.6.3.1. Técnicas de agrupamento de clientes

O aprendizado federado com o agrupamento de clientes é uma subárea da personalização de modelos, cujo objetivo é mitigar o problema da heterogeneidade de dados ou

capacidades computacionais. Para isso, os clientes são separados em diferentes grupos de treinamento. Esses grupos podem ser independentes entre si ou compartilhar informações dos modelos treinados de acordo com cada proposta.

Estratégias baseadas em agrupamento (*clustering*) procuram reunir dispositivos com características semelhantes para otimizar a tomada de decisão. As métricas utilizadas para formar esses grupos geralmente envolvem os pesos do modelo local, o tempo de treinamento por época, a distribuição dos dados e a localização geográfica [Mayhoub e M. Shami, 2024].

O TiFL [Chai et al., 2023] é uma estratégia de seleção que consiste em traçar um perfil dos clientes de acordo com métricas de latência e classificá-los em *tiers*. Em seguida, clientes são selecionados aleatoriamente dentro de cada *tier*.

Os autores em [Bao et al., 2021] consideram um cenário onde veículos se comunicam entre si e com unidades de beira de estrada (*roadside units* – RSUs). Os autores propõem um método para identificar clientes de borda em potencial para realizar aprendizado federado. Esses clientes coletam dados de sensores de outros veículos não selecionados ao seu entorno e treinam uma época não apenas com os dados gerados pelo próprio veículo, mas também com os dados coletados de clientes menos capazes em seu entorno. Os clientes aptos para realizar o treinamento são selecionados a partir de um cálculo de competência utilizando lógica *fuzzy*, considerando a estabilidade de sua conexão, a topologia da rede e um fator de conectividade. Dessa forma, a proposta busca diminuir a sobrecarga de comunicação, selecionando os clientes mais aptos.

[de Souza et al., 2024a] propõem um sistema de aprendizado federado no qual os clientes são agrupados baseando-se na similaridade dos dados. Para manter a privacidade dos dados, os pesos dos modelos treinados a partir do mesmo estado inicial são utilizados como um parâmetro de similaridade. Cada grupo treina um único modelo por meio do FedAVG. Essa abordagem aumenta a acurácia do modelo e reduz o tempo de convergência. No entanto, a proposta é limitada a tarefas específicas, como classificar dados semelhantes aos dados de treinamento do grupo, dificultando a generalização para amostras de outras distribuições fora do grupo. Assim, o ATHENA-FL [de Souza et al., 2024b] aprimora o sistema, fornecendo uma maneira de combinar os modelos gerados em diferentes grupos por meio do modelo Um-Contra-Todos (*One-versus-All* – OvA). Este modelo é formado por um conjunto de modelos baseados em detectores binários para cada classe, permitindo adição de novos detectores uma vez que haja novas classes.

Entretanto, as divisões dos clientes em grupos podem ser complexas em cenários reais, nos quais os cliente não possuem apenas um subconjunto das classes. Assim, a alocação de um cliente em um grupo incorreto pode ser altamente prejudicial tanto para o grupo quanto para o cliente. Dessa forma, a personalização de camadas da rede resolve a sensibilidade da personalização ao algoritmo de agrupamento ao permitir a pernalização de camadas ao invés do modelo completo.

2.6.3.2. Personalização por divisão arquitetural

Uma das estratégias de personalização mais utilizadas consiste em dividir a arquitetura do modelo em componentes globais e locais. O FedPer [Arivazhagan et al., 2019] é um dos trabalhos pioneiros nessa linha, propondo que as camadas iniciais do modelo, responsáveis pela extração de características gerais, sejam treinadas de forma federada, enquanto as camadas finais, voltadas à personalização, permaneçam exclusivamente locais.

Expandindo essa ideia, o FedRep [Collins et al., 2021] propõe um esquema de treinamento alternado que desacopla o aprendizado da representação global da tarefa de classificação local. Nesse procedimento, o cliente inicialmente congela o extrator de características recebido do servidor e treina apenas o classificador local, de modo a alinhar as camadas finais às particularidades de suas predições. Em seguida, o classificador é congelado, e o treinamento local prossegue atualizando apenas os pesos do extrator de características. Ao final das épocas locais, somente os pesos atualizados do extrator são enviados para a agregação federada, enquanto os classificadores permanecem nos clientes. Esse processo em duas etapas favorece o aprendizado de um extrator de características mais genérico e robusto, ao mesmo tempo em que preserva a especialização dos classificadores para as características e classes de cada nó.

O FedBABU [Oh et al., 2022] simplifica essa dinâmica ao mostrar que a atualização contínua do classificador durante o treinamento federado pode, em alguns casos, prejudicar a qualidade das representações globais aprendidas. Diferentemente do FedRep, essa abordagem mantém a cabeça de classificação fixa ao longo de todas as rodadas de comunicação, concentrando a atualização e a agregação exclusivamente no extrator de características. A personalização ocorre apenas após a convergência do modelo global, por meio de um ajuste fino final do classificador com os dados locais de cada veículo, o que pode resultar em maior precisão em cenários com elevada heterogeneidade de rótulos.

Apesar da personalização de camadas aportar benefícios para o aprendizado federado, a definição das camadas que devem ser personalizadas é uma tarefa complexa. Assim, uma forma de contornar a escolha das camadas é realizar a personalização no objetivo local de otimização.

2.6.3.3. Personalização por modificação do objetivo de otimização

Outra linha de personalização em aprendizado federado baseia-se na modificação do objetivo de otimização, de forma a favorecer a adaptação local ou a equilibrar explicitamente os objetivos globais e locais.

O pFedMe [T Dinh et al., 2020], por sua vez, também desacopla a atualização do modelo global da otimização do modelo personalizado, mas o faz por meio de uma formulação de otimização bi-nível, na qual a especialização local e a evolução global são tratadas de forma interdependente. Nessa abordagem, o problema é organizado em dois níveis. No nível interno, cada cliente busca os parâmetros mais adequados para seu modelo personalizado por meio de múltiplas iterações locais, utilizando um termo de proximidade para evitar desvios excessivos em relação à referência global. No nível externo, o

servidor atualiza o modelo global não a partir de gradientes brutos dos dados locais, mas com base nos pesos já otimizados dos modelos personalizados. Essa formulação permite integrar de forma mais consistente a personalização local ao processo de evolução global do sistema.

O Per-FedAvg [Fallah et al., 2020], por exemplo, reformula o objetivo do aprendizado federado ao tratá-lo como um problema de preparação para adaptação. Em vez de buscar um modelo que seja apenas com um desempenho aceitável para todos os clientes simultaneamente, o método procura aprender um modelo global que funcione como um ponto de partida ideal para rápida adaptação local. Tecnicamente, isso é realizado em duas etapas em cada nó. Primeiro, executa-se uma atualização provisória para avaliar como o modelo se comportaria após um breve ajuste com os dados locais. Em seguida, os parâmetros originais são otimizados de modo que essa versão adaptada apresente o melhor desempenho possível. Assim, o objetivo deixa de ser a simples minimização da perda do modelo atual e passa a considerar a perda do modelo após um passo de gradiente voltado à adaptação. Como resultado, obtém-se um modelo global mais flexível, capaz de se transformar em uma solução especializada com baixo custo computacional, exigindo poucas iterações de ajuste fino para se adequar ao domínio de cada cliente.

Dentro desse mesmo paradigma, o Ditto [Li et al., 2021d] propõe decompor o problema de otimização em duas frentes complementares: uma global e outra local. A otimização global consolida a contribuição de todos os clientes para o treinamento de um modelo comum, seguindo o fluxo tradicional do FedAvg. Em paralelo, a otimização local ajusta o modelo individual de cada cliente com base em seus próprios dados, incorporando à função de perda um termo de regularização que penaliza a distância euclidiana entre o modelo personalizado do cliente e o modelo global mais recente. Esse mecanismo permite que o modelo local aprenda padrões específicos do cliente sem se afastar excessivamente do conhecimento global, reduzindo o risco de sobreajuste e mantendo os benefícios das representações compartilhadas entre os participantes.

Entretanto, a personalização local mitiga apenas o problema de desempenho dos modelos causados pela diferença estatística dos clientes. Assim, estratégias recentes de personalização propõem a combinação de técnicas como destilação de conhecimento e modelos generativos para permitir o treinamento de modelos heterogêneos nos clientes que compartilham o conhecimento global.

2.6.3.4. Estratégias recentes de personalização

Uma tendência recente para reduzir drasticamente a sobrecarga de comunicação em redes intermitentes é a personalização baseada em *prompt learning*. O FedPrompt [Zhao et al., 2023] introduz esse conceito ao manter o modelo base (*backbone*) completamente congelado durante o treinamento federado. Em vez de transmitir gradientes associados a milhões de parâmetros, os clientes otimizam e compartilham apenas pequenos vetores de contexto, denominados *prompts*, que adaptam o modelo global às tarefas locais. Expandindo essa ideia, o FedMGP (*Federated Learning with Multi-Group Text-Visual Prompts*) [Bo et al., 2025] propõe o uso de *prompts* multigrupo para lidar com a variação semântica em dados visuais. O principal diferencial dessa abordagem é sua ca-

pacidade de personalizar o reconhecimento de padrões de acordo com o contexto, como condições climáticas ou sinalizações regionais específicas, sem a necessidade de reentreinar camadas mais pesadas da rede neural, o que torna a adaptação mais leve e eficiente para unidades com recursos limitados.

O FedTP (*Federated Learning by Transformer Personalization*) [Li et al., 2023] busca lidar com a heterogeneidade dos clientes por meio da personalização de modelos baseados em *Transformers*. O objetivo central da proposta é aprender uma base de transformação capaz de alinhar as representações dos dados locais de diferentes clientes, permitindo que cada nó personalize camadas específicas do modelo, como os mecanismos de atenção, para mitigar problemas de convergência. Tecnicamente, a abordagem emprega parâmetros de projeção aprendidos localmente para transformar o espaço de características, com o intuito de tornar mais semelhantes os dados provenientes de clientes distintos antes da agregação. No entanto, essa estratégia impõe uma sobrecarga de comunicação significativa, além de exigir tempo elevado para o ajuste dos parâmetros de projeção e para a convergência do modelo, o que limita sua aplicabilidade em cenários veiculares de alta mobilidade, nos quais as janelas de conexão tendem a ser curtas.

Em um nível mais estrutural de adaptação, a Mistura de Especialistas Personalizada (*Personalized Federated Learning with Mixture of Experts* – PFL-MoE) surge como uma alternativa promissora para lidar simultaneamente com a heterogeneidade de hardware e com a diversidade dos domínios de condução. O FedMoEKD (*Mixture of Specialized Experts for Model-Heterogeneous Personalized Federated Learning*) [Liang et al., 2025] propõe uma arquitetura esparsa na qual apenas uma fração dos parâmetros é ativada para cada entrada. Por meio de um mecanismo de *gating* aprendido localmente, cada cliente identifica quais submodelos são mais adequados ao seu cenário, processando apenas os componentes necessários. De forma complementar, o pFedMoE [Yi et al., 2026] introduz uma distinção entre especialistas globais e privados. Enquanto os especialistas globais consolidam o conhecimento compartilhado pela frota, os especialistas privados permanecem no cliente para aprender comportamentos específicos e padrões sensoriais mais sensíveis, promovendo uma personalização mais profunda sem comprometer a privacidade dos dados locais.

Em conjunto, essas abordagens mostram que a personalização em aprendizado federado pode ser realizada em diferentes níveis, desde ajustes na arquitetura e no objetivo de otimização até mecanismos mais recentes, como *prompts*, *Transformers* personalizados e mistura de especialistas (MoE). Em redes veiculares, essa pluralidade de soluções é essencial para acomodar a diversidade de contextos de operação da frota, incluindo diferenças de ambiente, comportamento de condução, condições climáticas, disponibilidade de conectividade e capacidade computacional dos veículos. Dessa forma, a personalização deixa de ser apenas uma alternativa para melhorar a acurácia local e passa a representar um componente central para viabilizar sistemas federados mais robustos, adaptativos e eficientes nesse domínio.

O FedCG (*Federated learning method that leverages Conditional Generative adversarial networks*) [Wu et al., 2021b] decompõe os modelos dos clientes em um extrator de características e um classificador, ao mesmo tempo que introduz uma CGAN (*Conditional Generative Adversarial Network*) dividida em um gerador e um discriminador. A

proposta compartilha apenas o classificador e o gerador com o servidor, que são agregados por meio de KD em um modelo global, compartilhado com os clientes. O objetivo é manter o extrator de características, constituído pelas camadas iniciais da rede neural do cliente, privado para evitar vazamento de dados durante o treinamento do modelo global e ao mesmo tempo reduzir o efeito de dados negativo da heterogeneidade de dados.

O CompFL (**Co**operative multiple **m**odel training framework for **p**ersonalized **F**ederated **L**earning) [Xu et al., 2024a] executa aprendizado federado com modelos heterogêneos. Em vez de treinar apenas um único modelo, os clientes com maior poder computacional também têm a probabilidade de selecionar um modelo mais simples para ajudar os clientes com menor poder computacional durante as rodadas de treinamento. Além disso, os clientes executam KD para aprender com diferentes modelos em suas amostras locais. No entanto, a proposta sobrecarrega os clientes com maior poder computacional, o que implica um alto custo computacional para contribuir com o sistema.

2.6.4. Utilização de múltiplos modelos de inteligência artificial

Além das estratégias de personalização, que visam reduzir os efeitos da heterogeneidade de dados, recentemente, existe a tendência de utilizar múltiplos modelos para mitigar a heterogeneidade de dispositivos. A fim de criar um modelo único, técnicas de agregação que utilizam a distilação do conhecimento têm sido aplicadas.

O FedGen (*Federated Distillation via Generative Learning*) [Zhu et al., 2021b] aborda o problema da distribuição heterogênea de dados. Os clientes compartilham a distribuição de rótulos com o servidor para treinar o gerador. Em seguida, o gerador é compartilhado com os clientes, para que possam executar a destilação de conhecimento.

FedAdKD (*Federated Adaptive Knowledge Distillation*) [Song et al., 2024] aplica a destilação de conhecimento para reduzir os efeitos da heterogeneidade das distribuições de dados dos clientes. Inicialmente, os clientes executam uma fase de treinamento federado com o FedAvg de um modelo de difusão (*diffusion model*), utilizado para gerar dados sintéticos. Dessa forma, o servidor pode ter acesso ao gerador que cria amostras sintéticas que seguem as distribuições sem revelar dados locais dos clientes. Após essa etapa, o servidor inicia o treinamento do modelo para a tarefa de aprendizado principal em conjunto com os clientes. A diferença dessa estratégia para o treinamento com o FedAvg é que o servidor, após agregar os modelos locais em um novo modelo global, realiza uma destilação de conhecimento com múltiplos professores (*Multi-Teacher Knowledge Distillation* – MTKD). Nesse caso, o modelo global após a agregação é utilizado como estudante dos modelos locais recebidos, que atuam como professores, usando o conjunto de dados sintético. A proposta mantém o uso de arquiteturas idênticas para realizar a agregação antes da destilação do conhecimento.

FedZIO (*Federated Distillation with Zonal Interaction Optimizer*) [Palazzo et al., 2023] é uma proposta de aprendizado federado descentralizado que utiliza a destilação de conhecimento como uma forma de treinamento dos modelos. Os autores assumem que existem no sistema clientes heterogêneos, que treinam um modelo adequado às suas capacidades computacionais. Por meio de um protocolo distribuído, os clientes recebem a predição (*logits*) de outros clientes para realizar a destilação de conhecimento. Entretanto, a proposta possui um desempenho limitado em

comparação com outras técnicas do estado da arte.

O HeteroFL [Diao et al., 2020] é uma das primeiras estratégias de FL que permite aos clientes treinar com modelos heterogêneos. A proposta divide o modelo global em submodelos e os clientes treinam o submodelo mais adequado à sua capacidade computacional. Como os submodelos estão relacionados ao modelo global, o esquema de agregação dispensa a destilação de conhecimento e o uso de dados públicos.

O FedDKC (*Federated Distillation algorithm based on Distributed Knowledge Congruence*) [Wu et al., 2024b] é um algoritmo que utiliza o KD para mitigar o problema decorrente da heterogeneidade no canal de comunicação e na distribuição de clientes no aprendizado federado. Os autores propõem uma estratégia para refinar o KD e reduzir a discrepância entre os clientes participantes do treinamento federado. Apesar da abordagem dispensar o uso de dados dos clientes para realizar o KD, os clientes devem realizar o KD. Assim, essa abordagem adiciona sobrecarga no lado do cliente.

O Felo [Chan e Ngai, 2025] difere de outras propostas FL-KD por utilizar características de nível intermediário, além dos *logits*, para transferir conhecimento de clientes com modelos heterogêneos. As características de nível intermediário são os resultados das camadas intermediárias do modelo. Uma vez gerados no lado do cliente, os *logits* e as características de nível intermediário são calculados localmente por classe e, em seguida, enviados juntamente com os parâmetros do modelo. O servidor agrega modelos com a mesma arquitetura de rede neural e, em cada classe, calcula a média das informações enviadas pelos clientes. Os autores estendem a primeira proposta com o Velo, que aprimora o processo ao adicionar um gerador para produzir dados sintéticos no lado do servidor, utilizando características de nível intermediário. No entanto, o lado do cliente apresenta uma sobrecarga em comparação com a proposta FedAvg, uma vez que os clientes precisam gerar logits e características de nível intermediário ao executar o cálculo da média desses dados.

O FedBKD (*Federated Bidirectional Knowledge Distillation*) [Qi et al., 2022] é um arcabouço de FL para lidar com a heterogeneidade de dados e de modelos em sistemas de borda em dispositivos IoT. Em uma fase de aquecimento, os clientes treinam um Autoencoder Variacional Condicional (*Conditional Variational Autoencoder* – CVAE) para gerar dados sintéticos no servidor. Durante o treinamento, o FedBKD utiliza a destilação de conhecimento em duas fases. Na fase de enlace ascendente (*uplink*), o servidor recebe os modelos dos clientes e executa um MTKD para treinar o modelo global. Além disso, na fase de enlace descendente (*downlink*), o modelo global obtido na fase anterior é usado localmente pelo cliente como professor para atualizar o modelo local. Apesar de a proposta ser projetada para permitir que os clientes treinem com diferentes arquiteturas de modelo, os experimentos testam a eficácia apenas com clientes compartilhando modelos similares, como o mesmo modelo base modificando apenas o número de camadas. Além disso, o FedBKD implica uma sobrecarga para o cliente, uma vez que na fase de enlace descendente a destilação deve ocorrer no dispositivo do cliente.

O FedAgg (*Agglomerative Federated Learning*) [Wu et al., 2024a] é uma proposta que aplica a destilação de conhecimento em uma arquitetura hierárquica fim-borda-nuvem para aprender um modelo conjunto em cada nível. Dispositivos de nuvem e de borda possuem mais poder computacional do que os nós finais. Portanto, aplicar um modelo homo-

gêneo a todos os nós subutiliza os recursos dos nós mais poderosos e reduz a acurácia do modelo global. Assim, os autores propõem o uso de modelos diferentes, de acordo com o poder de processamento, para melhor aproveitar os recursos do sistema. Como apenas os clientes possuem dados, o FedAgg compartilha amostras de ponte (bridge samples), que são amostras transformadas para preservar a privacidade dos usuários, para executar a destilação de conhecimento e treinar modelos nos nós de borda e de nuvem. No entanto, o FedAgg apresenta desvantagens para o cenário veicular, pois não é adaptado a um ambiente dinâmico. Por exemplo, a proposta pressupõe que os nós seguem uma distribuição fixa, de modo que o sistema sempre possui um único nó de nuvem e que os clientes são mais numerosos do que os nós de borda. Além disso, os autores assumem que os clientes são homogêneos e a heterogeneidade computacional do sistema encontra-se nos níveis hierárquicos da arquitetura, os quais não possuem amostras de dados.

O *Data-freE oNe-Shot federated lEarning* (DENSE) [Zhang et al., 2022a] treina modelos heterogêneos com apenas uma rodada de comunicação. Na proposta, os clientes treinam seus modelos, enquanto o servidor primeiramente treina um gerador com base nos modelos recebidos, garantindo similaridade, transferibilidade e estabilidade de forma adversarial. Em seguida, o servidor utiliza a destilação de conhecimento com os dados gerados para obter um modelo global. No entanto, o DENSE não considera as mudanças na distribuição de dados (*data shifts*) ao modelar o problema, o que leva o modelo global a esquecer catastroficamente conhecimentos úteis aprendidos em rodadas anteriores e a sofrer com a degradação de desempenho. O DFRD (Data-Free Robust Distillation) [Luo et al., 2023] é um método para Aprendizado Federado (FL) heterogêneo baseado em um gerador condicional treinado de forma adversarial. O sistema recebe os modelos dos clientes e os utiliza para treinar o gerador a fim de produzir amostras, garantindo, ao mesmo tempo, a fidelidade, a transferibilidade e a diversidade dos dados. Para evitar o esquecimento catastrófico, o DFRD utiliza uma cópia de média móvel exponencial (EMA) do gerador para armazenar o conhecimento prévio dos modelos locais. O gerador é então usado para criar dados sintéticos a cada rodada para realizar a destilação de conhecimento e melhorar o desempenho do modelo global. Além disso, o DFRD lida com a heterogeneidade de modelos empregando um método de treinamento parcial (PT) [Yang et al., 2022], no qual os clientes treinam submodelos de larguras diferentes extraídos do modelo global de acordo com seus orçamentos de recursos, o que significa que apenas um subconjunto dos pesos do modelo é atualizado localmente.

As estratégias de personalização permitem um treinamento mais especializado do modelo, aumentando o desempenho local. No entanto, essas estratégias são ortogonais às necessidades de aplicações federada veiculares, como a existência de dados rotulados. Assim, surgem novos paradigmas de aprendizado que buscam resolver problemas mais amplos, relacionados às aplicações.

2.6.5. Paradigma de aprendizado

Um dos principais desafios do aprendizado veicular é a obtenção de dados rotulados em aplicações como a visão computacional supervisionada. Isso ocorre devido ao cenário crítico, onde o usuário deve manter atenção constante ao trajeto e dificilmente pode contribuir com a rotulação ativa de dados quando o veículo está parado. Dessa forma, surgem novos paradigmas de aprendizado a fim de contornar esses desafios.

Os autores em FedMatch [Jeong et al., 2021] propõem um novo método de aprendizado semissupervisionado com troca de informação entre clientes de aprendizado federado. O FedMatch ataca o problema da falta de rótulos nos clientes de aprendizado federado. O aprendizado semissupervisionado em FedMatch utiliza o conceito de regularização de consistência, onde o modelo é treinado para prever o mesmo rótulo de classe para uma amostra original e a mesma amostra com aumentações. No entanto, o FedMatch amplia esse conceito para o cenário federado, introduzindo uma nova função de perda de consistência intercliente, onde modelos auxiliares são enviados pelo servidor baseados na similaridade do modelo local. Ademais, o FedMatch decompõe os parâmetros do modelo em dois grupos, um para o aprendizado semissupervisionado e um para o supervisionado. Durante o treinamento, a otimização desses parâmetros ocorre de forma disjunta, ao aprender com dados rotulados, o modelo atualiza os parâmetros supervisionados enquanto congela os não supervisionados, enquanto ao aprender com dados não rotulados, o modelo atualiza os parâmetros não supervisionados mantendo os supervisionados congelados.

O RSCFed [Liang et al., 2022] considera dois cenários de aprendizado federado semissupervisionado. O primeiro consiste em clientes com dados totalmente rotulados sendo treinados em conjunto com clientes com dados não rotulados. O segundo consiste em clientes com dados parcialmente rotulados sendo treinados em conjunto. Assumindo um cenário não-IID, o RSCFed consiste em considerar os modelos locais como modelos ruidosos e extrair diversos modelos de consenso por meio de amostragem aleatória antes de agregá-los ao modelo global. O aprendizado federado com o RSCFed consiste em (1) amostrar aleatoriamente clientes locais; (2) atribuir o modelo global atual aos clientes selecionados como inicialização e realizar o treino local nesses clientes; (3) realizar agregação ponderada pela distância nos modelos treinados; (4) repetir as etapas anteriores para criar um conjunto de modelos de subconsenso; (5) agregar um novo modelo a partir desse conjunto de subconsensos para se tornar o próximo modelo global.

O FedSTO [Kim et al., 2023] é proposto como uma solução para o caso particularmente desafiador de treinamento federado semissupervisionado em que apenas o servidor apresenta dados rotulados (*labels-at-server*), enquanto os clientes dispõem exclusivamente de dados não rotulados e possivelmente não-IID. O método é estruturado em dois estágios. No primeiro estágio, denominado treinamento seletivo, o modelo é treinado inicialmente com os dados do servidor. Em seguida, o *backbone* do modelo é ajustado aos dados não rotulados dos clientes por meio de uma abordagem de minimização de entropia baseada na comparação entre pseudo-rótulos de uma rede aluna e outra rede professora, que tem seus pesos atualizados por meio de uma média móvel exponencial (*Exponential Moving Average - EMA*) dos pesos da rede aluna. Com o fim de garantir maior robustez ao problema de viés de características, discutido na subseção 2.4.2, os autores incorporam uma técnica de regularização por ortogonalidade que promove maior adaptabilidade do modelo a dados provenientes de domínios distintos.

Os autores em [Li et al., 2022b] atacam o problema de alocação de recursos em redes veiculares onde enlaces V2V devem reutilizar os canais celulares já alocados, decidindo de forma autônoma e descentralizada quais canais reutilizar e qual potência de transmissão adotar, sem interromper as operações celulares e evitando colisões com outros enlaces V2V. Os autores propõem uma solução utilizando aprendizado por reforço

profundo multi-agente federado. Cada par V2V treina um modelo *Dueling Double Deep Q Network* de forma federada para selecionar qual canal celular deve ser utilizado e selecionar o nível de potência de transmissão. A função de recompensa formulada tenta otimizar a taxa de transmissão total de todos os usuários celulares enquanto mantém os requisitos de *Quality of Service* dos usuários celulares e pares V2V. Os clientes recebem um modelo global, coletam observações locais e treinam localmente. Os parâmetros dos modelos locais são enviados ao servidor e um novo modelo local é obtido através de agregação com o FedAvg.

2.7. Considerações Finais

Este minicurso apresentou o aprendizado federado veicular e discutiu as principais características e desafios que o diferenciam do FL tradicional. As arquiteturas abordadas demonstram diversas possibilidades de implementação do VFL segundo as tecnologias de comunicação presentes no veículo. No VFL é comum utilizar estratégias de agregação hierárquicas. Dessa forma, as RSU podem ser utilizadas como pontos de agregação entre o cliente e o servidor de nuvem, reduzindo a quantidade de dados transmitidos no núcleo da rede e oferecendo maior desempenho para os dispositivos próximos. Outra possibilidade é a realização de treinamento par a par, na qual os clientes atuam como cliente e servidor de agregação ao mesmo tempo. Porém, os problemas de generalização do modelo global são agravados em cenários hierárquicos e distribuídos. Assim, a principal estratégia de agregação utilizada é a centralizada quando possível. Como o treinamento do FL é realizado em múltiplos dispositivos, este está sujeito aos problemas enfrentados por sistemas distribuídos. Dessa forma, é necessário definir estratégias de coordenação para determinar o momento no qual o modelo é agregado e em que os clientes são sincronizados com o modelo global. A estratégia síncrona é estrita com relação ao tempo, descartando clientes muito lentos quando ultrapassam o tempo limite de resposta. Ao mesmo tempo, o sincronismo impõe tempo ocioso aos dispositivos mais rápidos. Por outro lado, o assíncrono elimina o descarte de modelos, agregando sob demanda a cada vez que um cliente envia suas atualizações. Entretanto, a atualização frequente impõe maior discordância entre modelos treinados por clientes retardatários e clientes rápidos, chamada de obsolescência. Por fim, a estratégia semissíncrona utiliza elementos síncronos, como a definição de tempo limite e descarte de modelos muito antigos ao mesmo tempo que permite a contribuição de clientes retardatários em rodadas diferentes para reduzir o descarte de modelos. Entretanto, o SGD, o principal algoritmo de atualização dos parâmetros, é sensível ao estado do modelo. Logo, versões diferentes prejudicam a agregação, reduzindo o desempenho de soluções não síncronas.

Limitações de latência, cobertura e vazão na rede de acesso podem acarretar no descarte de modelos e atrasos no término das rodadas, introduzindo requisitos de tolerância a falhas no projeto das máquinas de estado do VFL. As camadas de protocolos devem ser selecionadas conforme as necessidades de sincronismo entre cliente e servidor, sobrecarga de controle e confiabilidade do canal de comunicação. Ferramentas de gerenciamento de contêineres, transferência de arquivos e controle de versão de *software* são combinadas para construir arquiteturas de VFL que incluem não só a troca de parâmetros do modelo, como também a troca de mensagens de controle entre cliente e servidor.

A seção prática demonstrou como modelar as características da arquitetura e in-

fraestrutura apresentada, permitindo aos participantes o entendimento da implementação do VFL. Inicialmente discutiu-se como modelar a heterogeneidade dos dados dos clientes e seus recursos computacionais. Além disso, foi implementado um modelo de mobilidade que permite a análise do ambiente veicular. Por fim, demonstrou-se como agregar esses elementos no arcabouço Flower para a simulação fim a fim do aprendizado federado veicular. Após a atividade prática, foram apresentadas as principais tendências de pesquisas atuais, que utilizam arcabouços, como o Flower, para o desenvolvimento e teste de técnicas inovadoras.

As tendências de pesquisa identificadas indicam que o aprendizado federado em redes veiculares evolui para abordagens cada vez mais adaptadas à heterogeneidade dos dados, dos dispositivos e das condições de comunicação, afastando-se de estratégias genéricas baseadas em seleção aleatória de clientes e em um único modelo global. Nesse contexto, destacam-se como direções principais a seleção inteligente de clientes, as novas estratégias de agregação, a personalização de modelos, o agrupamento de clientes e o uso de múltiplos modelos, com o objetivo de reduzir o tempo de convergência, aumentar a acurácia e tornar o treinamento mais robusto em cenários dinâmicos e não-IID. Além disso, observa-se um avanço de técnicas baseadas em destilação de conhecimento e de novos paradigmas de aprendizado voltados a restrições práticas das aplicações, como assimetria de recursos, mudanças de distribuição e escassez de rótulos. Em conjunto, essas tendências mostram que a viabilidade do aprendizado federado veicular depende, cada vez mais, de soluções que conciliem adaptação ao contexto, eficiência de comunicação e robustez operacional.

Apesar dos avanços recentes, ainda há desafios a serem superados para a adoção efetiva do aprendizado federado em redes veiculares. Sobretudo, devido à dificuldade de lidar simultaneamente com a alta mobilidade do domínio veicular, heterogeneidade estatística dos dados, assimetria de recursos computacionais e instabilidade de comunicação. Além disso, muitas propostas ainda dependem de pressupostos simplificados, como conectividade consistente, custo computacional adicional nos clientes ou distribuições de dados relativamente estáveis, limitando sua aplicabilidade em cenários veiculares reais.

Nesse sentido, surge como direção futura a oportunidade de desenvolver métodos mais contextuais e realistas, capazes de integrar mecanismos de seleção de clientes, estratégias de agregação e personalização mais leves e robustas e técnicas de agrupamento dinâmico sensíveis às mudanças do ambiente e abordagens com múltiplos modelos que preservem a eficiência do sistema. Além disso, pesquisas futuras podem explorar com maior profundidade cenários realistas de implantação, incluindo aprendizado contínuo diante de mudanças de distribuição, escassez de rótulos, bem como mecanismos de segurança, confiança e privacidade compatíveis com a alta dinamicidade das redes veiculares. Assim, o avanço da área dependerá da proposição de soluções que não apenas melhorem o desempenho dos modelos, mas que também sejam efetivamente implementáveis em ambientes veiculares reais, dinâmicos e de larga escala.

Agradecimentos

Este capítulo foi realizado com apoio do CNPq (310234/2025-5, 407304/2025-8, 402531/2025-6, 408255/2023-4, 405940/2022-0 e 309304/2021-0); da Coordena-

ção de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Código de Financiamento 001, 88887.954253/2024-00 e 88887.987121/2024-00; da FAPERJ (E-26/200.380/2023, E-26/204.122/2024 e E-26/210.778/2025); da FAPESP (2023/00673-7 e 2023/00811-0) e da Fundação de Desenvolvimento da Pesquisa - Fundep - Rota 2030 em conjunto dos nossos parceiros Stellantis e Mobway.

Referências

[Abad et al., 2020] Abad, M. S. H., Ozfatura, E., Gunduz, D. e Ercetin, O. (2020). Hierarchical Federated Learning across Heterogeneous Cellular Networks. Em *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 8866–8870. IEEE.

[Ali et al., 2025] Ali, A., Jianjun, H. e Jabbar, A. (2025). Recent Advances in Federated Learning for Connected Autonomous Vehicles: Addressing Privacy, Performance, and Scalability Challenges. *IEEE Access*.

[Amadeo et al., 2025] Amadeo, M., Campolo, C., Ruggeri, G. e Molinaro, A. (2025). Improving Communication Performance of Federated Learning: A Networking Perspective. *Computer Networks*, 267:111353.

[Arivazhagan et al., 2019] Arivazhagan, M. G., Aggarwal, V., Singh, A. K. e Choudhary, S. (2019). Federated Learning with Personalization Layers. *arXiv:1912.00818*.

[Azimi-Abarghouyi e Fischione, 2025] Azimi-Abarghouyi, S. M. e Fischione, C. (2025). Multi-Layer Hierarchical Federated Learning with Quantization. *arXiv:2505.08145*.

[Bao et al., 2021] Bao, W., Wu, C., Guleng, S., Zhang, J., Yau, K.-L. A. e Ji, Y. (2021). Edge Computing-Based Joint Client Selection and Networking Scheme for Federated Learning in Vehicular IoT. *China Communications*, 18(6):39–52.

[Beltrán et al., 2023] Beltrán, E. T. M. et al. (2023). Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges. *IEEE Communications Surveys & Tutorials*, 25(4):2983–3013.

[Beutel et al., 2020] Beutel, D. J. et al. (2020). Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*.

[Bo et al., 2025] Bo, W., Sun, Y., Wang, Y., Zhang, X. e Li, Z. (2025). FedMGP: Personalized Federated Learning with Multi-Group Text-Visual Prompts. Em *Conference on Neural Information Processing Systems*.

[Buyukates e Ulukus, 2021] Buyukates, B. e Ulukus, S. (2021). Timely Communication in Federated Learning. Em *International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, p. 1–6. IEEE.

[Chahoud et al., 2023a] Chahoud, M., Otoum, S. e Mourad, A. (2023a). On the Feasibility of Federated Learning Towards On-demand Client Deployment at the Edge. *Information Processing & Management*, 60(1):103150.

[Chahoud et al., 2023b] Chahoud, M., Sami, H., Mourad, A., Otoum, S., Otrok, H., Bentahar, J. e Guizani, M. (2023b). On-demand-FL: A Dynamic and Efficient Multicriteria Federated Learning Client Deployment Scheme. *IEEE Internet of Things Journal*, 10(18):15822–15834.

[Chahoud et al., 2025] Chahoud, M., Sami, H., Mourad, A., Otrok, H., Bentahar, J. e Guizani, M. (2025). On-demand model and client deployment in federated learning with deep reinforcement learning. *IEEE Internet of Things Journal*.

[Chai et al., 2023] Chai, Z. et al. (2023). TiFL: A Tier-based Federated Learning System. Em *International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '20, p. 125–136. Association for Computing Machinery.

[Chan e Ngai, 2025] Chan, Y.-H. e Ngai, E. C. (2025). Exploiting Features and Logits in Heterogeneous Federated Learning. *Computer Networks*, 264:111271.

[Chatzoulis et al., 2023] Chatzoulis, D. et al. (2023). 5G V2X Performance Comparison for Different Channel Coding Schemes and Propagation Models. *Sensors*, 23(5):2436.

[Chellapandi et al., 2023] Chellapandi, V. P., Yuan, L., Brinton, C. G., Żak, S. H. e Wang, Z. (2023). Federated Learning for Connected and Automated Vehicles: A Survey of Existing Approaches and Challenges. *IEEE Transactions on Intelligent Vehicles*, 9(1):119–137.

[Chen et al., 2023a] Chen, M., Xu, Y., Xu, H. e Huang, L. (2023a). Enhancing Decentralized Federated Learning for Non-IID Data on Heterogeneous Devices. Em *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, p. 2289–2302. IEEE.

[Chen et al., 2024] Chen, Z., Ni, Z., Guan, P., Wang, L., Cai, L. X., Hashemi, M. e Li, Z. (2024). Optimizing NOMA Transmissions to Advance Federated Learning in Vehicular Networks. Em *Global Communications Conference (GLOBECOM)*, p. 962–967.

[Chen et al., 2023b] Chen, Z., Shi, L., Liu, X., Ai, X., Liu, S. e Xu, Y. (2023b). Boosting Distributed Machine Learning Training Through Loss-tolerant Transmission Protocol. Em *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, p. 1–10. ISSN: 2766-8568.

[Cleland et al., 2022] Cleland, G., Wu, D., Ullah, R. e Varghese, B. (2022). FedComm: Understanding Communication Protocols for Edge-based Federated Learning. Em *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*, p. 71–81, Vancouver, WA, USA. IEEE.

[Collins et al., 2021] Collins, L., Hassani, H., Mokhtari, A. e Shakkottai, S. (2021). Exploiting Shared Representations for Personalized Federated Learning. Em *International Conference on Machine Learning*, p. 2089–2099. PMLR.

[De Rango et al., 2021] De Rango, F., Guerrieri, A., Raimondo, P. e Spezzano, G. (2021). A Novel Edge-based Multi-Layer Hierarchical Architecture for Federated Learning. Em *DASC/PiCom/CBDCom/CyberSciTech*, p. 221–225. IEEE.

[de Souza et al., 2024a] de Souza, L. A. C., Camilo, G. F., Campista, M. E. M. e Costa, L. H. M. (2024a). Increasing the Accuracy of Federated Learning on Non-IID Scenarios using Client Clustering. Em *Latin-American Conference on Communications (LATINCOM)*, p. 1–6. IEEE.

[de Souza et al., 2024b] de Souza, L. A. C., Camilo, G. F., Rebello, G. A. F., Sammarco, M., Campista, M. E. M. e Costa, L. H. M. K. (2024b). ATHENA-FL: Avoiding Statistical Heterogeneity with One-versus-All in Federated Learning. *Journal of Internet Services and Applications*, 15(1):273–288.

[de Souza et al., 2026] de Souza, L. A. C. et al. (2026). CAIROS: Controle Adaptativo do Aprendizado Federado em Redes Sem Fio. Em *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. SBC.

[de Souza et al., 2025] de Souza, L. A. C., Sammarco, M., Achir, N., Costa, L. H. M. K. e Campista, M. E. M. (2025). TOFL: Time Optimized Federated Learning. Em *SBSeg 2025-Simpósio Brasileiro de Cibersegurança*.

[Diao et al., 2020] Diao, E., Ding, J. e Tarokh, V. (2020). HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. *arXiv preprint arXiv:2010.01264*.

[EletronicsMaker, 2024] EletronicsMaker (2024). Driving the Future: The Role of Computing Power in Autonomous Vehicles. https://electronicsmaker.com/driving-the-future-the-role-of-computing-power-in-autonomous-vehicles.

[Fallah et al., 2020] Fallah, A., Mokhtari, A. e Ozdaglar, A. (2020). Personalized Federated Learning: A Meta-Learning Approach. *arXiv preprint arXiv:2002.07948*.

[Ferguson, 1973] Ferguson, T. S. (1973). A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, p. 209–230.

[Fittipaldi et al., 2025] Fittipaldi, G., Couto, R. S. e Costa, L. H. (2025). Exploring Traffic Pattern Variability in Vehicular Federated Learning. *Computer Communications*, p. 108279.

[Fu et al., 2023] Fu, L., Zhang, H., Gao, G., Zhang, M. e Liu, X. (2023). Client Selection in Federated Learning: Principles, Challenges, and Opportunities. *IEEE Internet of Things Journal*, 10(24):21811–21819.

[Galende et al., 2024] Galende, B. A. et al. (2024). FLIP: A New Approach for Easing the Use of Federated Learning. *Applied Sciences*.

[Ghanem et al., 2022] Ghanem, M., Dawoud, F., Gamal, H., Soliman, E., El-Batt, T. e Sharara, H. (2022). FLoBC: A Decentralized Blockchain-Based Federated Learning Framework. Em *International Conference on Blockchain Computing and Applications (BCCA)*, p. 85–92. IEEE.

[Han et al., 2024] Han, D.-J., Hosseinalipour, S., Love, D. J., Chiang, M. e Brinton, C. G. (2024). Cooperative Federated Learning Over Ground-to-Satellite Integrated Networks: Joint Local Computation and Data Offloading. *IEEE Journal on Selected Areas in Communications*, 42(5):1080–1096.

[He et al., 2020] He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Zhu, X., Wang, J., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M. e Avestimehr, S. (2020). FedML: A Research Library and Benchmark for Federated Machine Learning. arXiv:2007.13518 [cs].

[Heydarishirayeh, 2023] Heydarishirayeh, N. (2023). *FedMingle: Communication in Federated Learning*. PhD thesis, Politecnico di Milano.

[Huang et al., 2021] Huang, Y. et al. (2021). Evaluating Gradient Inversion Attacks and Defenses in Federated Learning. Em Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. e Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, p. 7232–7241. Curran Associates, Inc.

[Huedo et al., 2025] Huedo, E. et al. (2025). On-Demand Deployment of Edge Cloud Infrastructures for Federated Learning. *Software: Practice and Experience*, 55(8):1377–1388.

[Islam et al., 2025] Islam, M. S., Panta, S., Xu, F., Yuan, X., Chen, L. e Tzeng, N.-F. (2025). SEAFL: Enhancing Efficiency in Semi-Asynchronous Federated Learning through Adaptive Aggregation and Selective Training. Em *International Parallel and Distributed Processing Symposium (IPDPS)*, p. 509–519. IEEE.

[Jeong e Kountouris, 2023] Jeong, E. e Kountouris, M. (2023). Personalized Decentralized Federated Learning with Knowledge Distillation. Em *IEEE International Conference on Communications*, p. 1982–1987.

[Jeong et al., 2021] Jeong, W., Yoon, J., Yang, E. e Hwang, S. J. (2021). Federated Semi-Supervised Learning with Inter-Client Consistency & Disjoint Learning.

[Ji et al., 2024] Ji, S., Tan, Y., Saravirta, T., Yang, Z., Liu, Y., Vasankari, L., Pan, S., Long, G. e Walid, A. (2024). Emerging Trends in Federated Learning: from Model Fusion to Federated X Learning. *International Journal of Machine Learning and Cybernetics*, 15(9):3769–3790.

[Kairouz et al., 2021] Kairouz, P. et al. (2021). Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210.

[Kalra et al., 2023] Kalra, S., Wen, J., Cresswell, J. C., Volkovs, M. e Tizhoosh, H. R. (2023). Decentralized Federated Learning through Proxy Model Sharing. *Nature communications*, 14(1):2899.

[Kang et al., 2025] Kang, N., Lim, Y. e Im, J. (2025). Optimizing Federated Learning: Addressing Key Challenges in Real-World Applications. *Internet of Things Journal*.

[Karimireddy et al., 2020] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S. e Suresh, A. T. (2020). SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. Em *International conference on machine learning*, p. 5132–5143. PMLR.

[Kim et al., 2023] Kim, T., Lin, E., Lee, J., Lau, C. e Mugunthan, V. (2023). Navigating Data Heterogeneity in Federated Learning: A Semi-Supervised Federated Object Detection. Em Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M. e Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, p. 2074–2096. Curran Associates, Inc.

[Kornaros et al., 2020] Kornaros, G., Tomoutzoglou, O., Mbakoyiannis, D., Karadimitriou, N., Coppola, M., Montanari, E., Deligiannis, I. e Gherardi, G. (2020). Towards Holistic Secure Networking in Connected Vehicles through Securing CAN-Bus Communication and Firmware-Over-The-Air Updating. *Journal of Systems Architecture*, 109:101761.

[Lai et al., 2021] Lai, F., Zhu, X., Madhyastha, H. V. e Chowdhury, M. (2021). Oort: Efficient Federated Learning via Guided Participant Selection. Em *USENIX OSDI*, p. 19–35.

[Li et al., 2022a] Li, B., Jiang, Y., Pei, Q., Li, T., Liu, L. e Lu, R. (2022a). FEEL: Federated End-to-End Learning With Non-IID Data for Vehicular Ad Hoc Networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):16728–16740.

[Li et al., 2021a] Li, B., Jiang, Y., Sun, W., Niu, W. e Wang, P. (2021a). FedVANET: Efficient Federated Learning with Non-IID Data for Vehicular Ad Hoc Networks. Em *IEEE Global Communications Conference (GLOBECOM)*, p. 1–6. IEEE.

[Li et al., 2021b] Li, C., Li, G. e Varshney, P. K. (2021b). Decentralized Federated Learning via Mutual Knowledge Transfer. *IEEE Internet of Things Journal*, 9(2):1136–1147.

[Li et al., 2023] Li, H., Cai, Z., Wang, J., Tang, J., Ding, W., Lin, C.-T. e Shi, Y. (2023). FedTP: Federated Learning by Transformer Personalization. *IEEE Transactions on Neural Networks and Learning Systems*.

[Li et al., 2024] Li, J., Chen, T. e Teng, S. (2024). A Comprehensive Survey on Client Selection Strategies in Federated Learning. *Computer Networks*, 251:110663.

[Li et al., 2021c] Li, Q., He, B. e Song, D. (2021c). Model-Contrastive Federated Learning. Em *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 10713–10722.

[Li et al., 2021d] Li, T., Hu, S., Beirami, A. e Smith, V. (2021d). Ditto: Fair and Robust Federated Learning through Personalization. Em *International Conference on Machine Learning*, p. 6357–6368. PMLR.

[Li et al., 2020] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A. e Smith, V. (2020). Federated Optimization in Heterogeneous Networks. *Machine Learning and Systems*, 2:429–450.

[Li et al., 2022b] Li, X., Lu, L., Ni, W., Jamalipour, A., Zhang, D. e Du, H. (2022b). Federated Multi-Agent Deep Reinforcement Learning for Resource Allocation of Vehicle-to-Vehicle Communications. *IEEE Transactions on Vehicular Technology*, 71(8):8810–8824.

[Liang et al., 2025] Liang, T., Hu, M. e Sun, E. (2025). Mixture of specialized experts for model-heterogeneous personalized federated learning. *IEEE Networking Letters*.

[Liang et al., 2022] Liang, X., Lin, Y., Fu, H., Zhu, L. e Li, X. (2022). RSCFed: Random Sampling Consensus Federated Semi-Supervised Learning. Em *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 10154–10163.

[Liu et al., 2024a] Liu, J., Jia, J., Che, T., Huo, C., Ren, J., Zhou, Y., Dai, H. e Dou, D. (2024a). FedASMU: Efficient Asynchronous Federated Learning with Dynamic Staleness-aware Model Update. Em *Conference on Artificial Intelligence (AAAI)*, volume 38, p. 13900–13908.

[Liu et al., 2020] Liu, L., Zhang, J., Song, S. e Letaief, K. B. (2020). Client-Edge-Cloud Hierarchical Federated Learning. Em *IEEE International Conference on Communications (ICC)*, p. 1–6. IEEE.

[Liu et al., 2024b] Liu, Y., Shi, Y., Li, Q., Wu, B., Wang, X. e Shen, L. (2024b). Decentralized Directed Collaboration for Personalized Federated Learning. Em *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 23168–23178.

[Lopez et al., 2018] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P. e Wiessner, E. (2018). Microscopic Traffic Simulation using SUMO. Em *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, p. 2575–2582.

[Luo et al., 2022] Luo, B. et al. (2022). Tackling System and Statistical Heterogeneity for Federated Learning with Adaptive Client Sampling. Em *IEEE INFOCOM*, p. 1739–1748.

[Luo et al., 2023] Luo, K., Wang, S., Fu, Y., Li, X., Lan, Y. e Gao, M. (2023). DFRD: Data-Free Robustness Distillation for Heterogeneous Federated Learning. *Advances in Neural Information Processing Systems*, 36:17854–17866.

[Magdum et al., 2021] Magdum, S. S., Franklin, A. e Tamma, B. R. (2021). A Cooperative Federated Learning Mechanism for Collision Avoidance using Cellular and 802.11p based Radios Opportunistically. Em *2021 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, p. 284–289.

[Mayhoub e M. Shami, 2024] Mayhoub, S. e M. Shami, T. (2024). A Review of Client Selection Methods in Federated Learning. *Archives of Computational Methods in Engineering*, 31(2):1129–1152.

[McMahan et al., 2017] McMahan, B. et al. (2017). Communication-efficient Learning of Deep Networks from Decentralized Data. *Artificial Intelligence and Statistics*, p. 1273–1282.

[Moon et al., 2024] Moon, J., Yang, S. e Lee, K. (2024). FedOps: A Platform of Federated Learning Operations With Heterogeneity Management. *IEEE Access*, 12:4301–4314.

[Moon e Lim, 2024] Moon, S. e Lim, Y. (2024). Client Selection for Federated Learning in Vehicular Edge Computing: A Deep Reinforcement Learning Approach. *IEEE Access*, 12:131337–131348.

[Neto et al., 2022] Neto, H. N. C., Dusparic, I., Mattos, D. M. e Fernande, N. C. (2022). FedSA: Accelerating Intrusion Detection in Collaborative Environments with Federated Simulated Annealing. Em *International Conference on Network Softwarization (NetSoft)*, p. 420–428. IEEE.

[Nguyen et al., 2022] Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M. e Huba, D. (2022). Federated Learning with Buffered Asynchronous Aggregation. Em *International conference on artificial intelligence and statistics*, p. 3581–3607. PMLR.

[Nishio e Yonetani, 2019] Nishio, T. e Yonetani, R. (2019). Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. Em *International Conference on Communications*, p. 1–7.

[Oh et al., 2022] Oh, J., Kim, S. e Yun, S. Y. (2022). FedBABU: Toward Enhanced Representation for Federated Image Classification. Em *10th International Conference on Learning Representations, ICLR 2022*.

[Palazzo et al., 2023] Palazzo, L., Pennisi, M., Bellitto, G. e Kavasidis, I. (2023). FeDZIO: Decentralized Federated Knowledge Distillation on Edge Devices. Em *International Conference on Image Analysis and Processing*, p. 201–210. Springer.

[Posner et al., 2021] Posner, J., Tseng, L., Aloqaily, M. e Jararweh, Y. (2021). Federated learning in vehicular networks: Opportunities and solutions. *IEEE Network*, 35(2):152–159.

[Qi et al., 2022] Qi, P. et al. (2022). FedBKD: Heterogenous Federated Learning via Bidirectional Knowledge Distillation for Modulation Classification in IoT-Edge System. *IEEE Journal of Selected Topics in Signal Processing*, 17(1):189–204.

[Rai et al., 2022] Rai, S., Kumari, A. e Prasad, D. K. (2022). Client Selection in Federated Learning under Imperfections in Environment. *AI*, 3(1):124–145.

[Sabuhi et al., 2024] Sabuhi, M., Musilek, P. e Bezemer, C.-P. (2024). Micro-FL: A Fault-Tolerant Scalable Microservice-based Platform for Federated Learning. *Future Internet*, 16(3):70.

[Shi et al., 2023] Shi, Y., Shen, L., Wei, K., Sun, Y., Yuan, B., Wang, X. e Tao, D. (2023). Improving the model consistency of decentralized federated learning. Em *International Conference on Machine Learning*, p. 31269–31291. PMLR.

[Soltani et al., 2022] Soltani, B., Haghighi, V., Mahmood, A., Sheng, Q. Z. e Yao, L. (2022). A Survey on Participant Selection for Federated Learning in Mobile Networks. Em *Workshop on Mobility in the Evolving Internet Architecture*, p. 19–24. ACM.

[Song et al., 2024] Song, Y., Liu, H., Zhao, S., Jin, H., Yu, J., Liu, Y., Zhai, R. e Wang, L. (2024). FedAdKD: Heterogeneous Federated Learning via Adaptive Knowledge Distillation. *Pattern Analysis and Applications*, 27(4):134.

[Stallkamp et al., 2011] Stallkamp, J., Schlipsing, M., Salmen, J. e Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. Em *International Joint Conference on Neural Networks*, p. 1453–1460. IEEE.

[Stripelis et al., 2022] Stripelis, D., Thompson, P. M. e Ambite, J. L. (2022). Semi-synchronous federated learning for energy-efficient training and accelerated convergence in cross-silo settings. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(5):1–29.

[Su et al., 2024] Su, D. et al. (2024). Communication Cost-Aware Client Selection in Online Federated Learning: A Lyapunov Approach. *Computer Networks*, p. 110517.

[Sun et al., 2025] Sun, Y., Shen, L. e Tao, D. (2025). Towards understanding generalization and stability gaps between centralized and decentralized federated learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[T Dinh et al., 2020] T Dinh, C., Tran, N. e Nguyen, J. (2020). Personalized Federated Learning with Moreau Envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405.

[Thomaz et al., 2024] Thomaz, G. A., Ferreira, J. P. M. F., Barry, T., Sammarco, M. e Campista, M. E. M. (2024). UBOTA Protocol - UDP Bursts for Over-the-Air Secure Vehicular Software Updates. Em *2024 IEEE 13th International Conference on Cloud Networking (CloudNet)*, p. 1–9. ISSN: 2771-5663.

[Thomaz et al., 2025] Thomaz, G. A., Silva, F. D. d. M., Souza, L. A. C. d., Costa, L. H. M. K. e Campista, M. E. M. (2025). AGATA – Arquitetura para Gerenciamento Automático de Tarefas de Aprendizado Federado. Em *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, p. 121–128. SBC.

[Wang et al., 2023] Wang, E., Chen, B., Chowdhury, M., Kannan, A. e Liang, F. (2023). FLINT: A Platform for Federated Learning Integration. *Machine Learning and Systems*, 5:21–34.

[Wang et al., 2020a] Wang, H., Kaplan, Z., Niu, D. e Li, B. (2020a). Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. Em *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, p. 1698–1707.

[Wang et al., 2020b] Wang, J. et al. (2020b). Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. *NeurIPS*, 33:7611–7623.

[Wang et al., 2022] Wang, Z., Zhang, Z., Tian, Y., Yang, Q., Shan, H., Wang, W. e Quek, T. Q. (2022). Asynchronous federated learning over wireless communication networks. *IEEE Transactions on Wireless Communications*, 21(9):6961–6978.

[Wu et al., 2021a] Wu, W. et al. (2021a). SAFA: A Semi-Asynchronous Protocol for Fast Federated Learning with Low Overhead. *IEEE Transactions on Computers*, 70(5):655–668.

[Wu et al., 2021b] Wu, Y., Kang, Y., Luo, J., He, Y. e Yang, Q. (2021b). FedCG: Leverage Conditional GAN for Protecting Privacy and Maintaining Competitive Performance in Federated Learning. *arXiv preprint arXiv:2111.08211*.

[Wu et al., 2024a] Wu, Z., Sun, S., Wang, Y., Liu, M., Gao, B., Pan, Q., He, T. e Jiang, X. (2024a). Agglomerative Federated Learning: Empowering Larger Model Training via End-Edge-Cloud Collaboration. Em *IEEE Conference on Computer Communications (INFOCOM)*, p. 131–140. ISSN: 2641-9874.

[Wu et al., 2024b] Wu, Z., Sun, S., Wang, Y., Liu, M., Pan, Q., Zhang, J., Li, Z. e Liu, Q. (2024b). Exploring the Distributed Knowledge Congruence in Proxy-data-free Federated Distillation. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–34.

[Xiao et al., 2025] Xiao, S., Huang, X., Zhou, M., Liang, C. e Chen, Q. (2025). FedDLD: Dual-Level Federated Distillation with Adaptive Knowledge Transfer for DAG-secured IoVs. *IEEE Transactions on Vehicular Technology*.

[Xie et al., 2019] Xie, C., Koyejo, S. e Gupta, I. (2019). Asynchronous Federated Optimization. *arXiv e-prints*, p. arXiv:1903.03934.

[Xie et al., 2026] Xie, R., Liang, W., Diao, Z., Meng, X. e Li, K. (2026). Personalized hierarchical federated learning framework for the internet of vehicles based on split meta-learning. *IEEE Internet of Things Journal*.

[Xu et al., 2023] Xu, C., Qu, Y., Xiang, Y. e Gao, L. (2023). Asynchronous federated learning on heterogeneous devices: A survey. *Computer Science Review*, 50:100595.

[Xu et al., 2024a] Xu, J., Wan, S., Li, Y., Luo, S., Chen, Z., Shao, Y., Chen, Z., Huang, S.-L. e Song, L. (2024a). Cooperative Multi-Model Training for Personalized Federated Learning Over Heterogeneous Devices. *Journal of Selected Topics in Signal Processing*, 19(1):195–207.

[Xu et al., 2024b] Xu, Z., Jiang, F., Niu, L., Jia, J. e Poovendran, R. (2024b). Brave: Byzantine-Resilient and Privacy-Preserving Peer-to-Peer Federated Learning. arXiv:2401.05562 [cs].

[Yang et al., 2023] Yang, S., Moon, J., Kim, J., Lee, K. e Lee, K. (2023). FLScalize: Federated Learning Lifecycle Management Platform. *IEEE Access*, 11:47212–47222.

[Yang et al., 2022] Yang, T.-J., Guliani, D., Beaufays, F. e Motta, G. (2022). Partial Variable Training for Efficient On-Device Federated Learning. Em *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 4348–4352. IEEE.

[Ye e ohters, 2023] Ye, M. e ohters (2023). Heterogeneous Federated Learning: State-of-the-Art and Research Challenges. *ACM Comput. Surv.*, 56(3).

[Yi et al., 2026] Yi, L., Yu, H., Wang, G., Liu, X. e Hu, Q. (2026). pFedMoE: Data-Level Personalization With Mixture of Experts in Model-Heterogeneous Personalized Federated Learning. *IEEE Transactions on Knowledge and Data Engineering*.

[Yuan et al., 2024] Yuan, L., Wang, Z., Sun, L., Yu, P. S. e Brinton, C. G. (2024). Decentralized Federated Learning: A Survey and Perspective. *IEEE Internet of Things Journal*, 11(21):34617–34638.

[Zejun et al., 2024] Zejun, L. et al. (2024). Mitigating Straggler Effect in Federated Learning based on Reconfigurable Intelligent Surface over Internet of Vehicles. *China Communications*, 21(8):62–78.

[Zhang et al., 2022a] Zhang, J., Chen, C., Li, B., Lyu, L., Wu, S., Ding, S., Shen, C. e Wu, C. (2022a). DENSE: Data-Free One-Shot Federated Learning. *Advances in Neural Information Processing Systems*, 35:21414–21428.

[Zhang et al., 2022b] Zhang, J., Li, Z., Li, B., Xu, J., Wu, S., Ding, S. e Wu, C. (2022b). Federated Learning with Label Distribution Skew via Logits Calibration. Em Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G. e Sabato, S., editors, *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, p. 26311–26329. PMLR.

[Zhang et al., 2022c] Zhang, L. et al. (2022c). Fine-Tuning Global Model via Data-Free Knowledge Distillation for Non-IID Federated Learning. Em *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 10174–10183.

[Zhang et al., 2021] Zhang, P., Wang, C., Jiang, C. e Han, Z. (2021). Deep Reinforcement Learning Assisted Federated Learning Algorithm for Data Management of IIoT. *IEEE Transactions on Industrial Informatics*, 17(12):8475–8484.

[Zhang et al., 2023] Zhang, X. et al. (2023). Federated Learning-Assisted Vehicular Edge Computing: Architecture and Research Directions. *IEEE Vehicular Technology Magazine*, 18(4):75–84.

[Zhang et al., 2022d] Zhang, Y. et al. (2022d). S4OD: Semi-Supervised Learning for Single-Stage Object Detection.

[Zhao et al., 2023] Zhao, H., Du, W., Li, F., Li, P. e Liu, G. (2023). FedPrompt: Communication-Efficient and Privacy-Preserving Prompt Tuning in Federated Learning. Em *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 1–5. IEEE.

[Zhu et al., 2021a] Zhu, Q. et al. (2021a). 3GPP TR 38.901 Channel Model. Em *The Wiley 5G Ref: The Essential 5G Reference Online*, p. 1–35. Wiley Press Hoboken.

[Zhu et al., 2021b] Zhu, Z., Hong, J. e Zhou, J. (2021b). Data-Free Knowledge Distillation for Heterogeneous Federated Learning. Em *International Conference on Machine Learning (PMLR)*, p. 12878–12889. ISSN: 2640-3498.

Capítulo 3

Agentes Inteligentes para Configuração de Redes de Computadores: Da Teoria à Prática com LLM, SLM, RAG e IA Agentic

William L. Reiznautt (UNICAMP), Eduardo Cerqueira (UFPA),
Diogo M. da Cunha (UNICAMP), Leandro A. Villas (UNICAMP),
Antonio A. F. Loureiro (UFMG), Denis Rosário (UFPA),
Allan M. de Souza (UNICAMP), Nelson L. S. da Fonseca (UNICAMP)

Abstract

Network management has become increasingly complex due to the convergence of cloud, edge, 5G/O-RAN, and multivendor environments. This chapter presents intelligent agents for network configuration in the context of NetOps 2.0, integrating LLMs, SLMs, RAG, and Agentic AI. It discusses multimodel architectures, the SLM-first strategy, and reasoning paradigms such as ReAct, Pre-Act, and Structured Cognitive Loop. The MCP and A2A protocols are also examined, along with their security challenges. In the practical section, the chapter describes the implementation of an agent using Python, Ollama, and OpenWebUI for Linux network automation, including bridges, VLANs, IPv4/IPv6 routing, and VXLAN under the cycle Interpret→Plan→Execute→Validate→Correct. A case study and final benchmarks highlight paths toward autonomous networks.

Resumo

O gerenciamento de redes tornou-se complexo diante da convergência entre nuvem, borda, 5G/O-RAN e ambientes multivendedor. Este capítulo apresenta agentes inteligentes para configuração de redes no contexto do NetOps 2.0, integrando LLMs, SLMs, RAG e IA Agentic. São discutidas arquiteturas multimodelo, a estratégia SLM-first e paradigmas como ReAct, Pre-Act e Structured Cognitive Loop. Também são analisados os protocolos MCP e A2A e seus desafios de segurança. Na parte prática, descreve-se a implementação de um agente com Python, Ollama e OpenWebUI para automação de redes Linux, incluindo bridges, VLANs, roteamento IPv4/IPv6 e VXLAN no ciclo Interpretar→Planejar→Executar→Validar→Corrigir. Um estudo de caso e benchmarks finais apontam caminhos para redes autônomas.

3.1. Introdução

As redes de telecomunicações evoluíram para sistemas de larga escala altamente complexos, abrangendo desde o núcleo da rede (*core*) e o transporte até a borda (*edge*) e as redes de acesso via rádio (RAN) [Zhou et al. 2024]. Essa infraestrutura moderna é caracterizada por ambientes híbridos que integram recursos físicos (*bare metal*), virtualização de funções de rede (*Network Function Virtualization* – NFV), contêineres e Redes Definidas por Software (*Software Defined Network* – SDN) [Huang et al. 2023].

A natureza heterogênea e multivendedor dessas redes exige que os operadores gerenciem uma ampla variedade de dispositivos de diferentes fornecedores, cada um com suas próprias linguagens de configuração, protocolos e manuais técnicos [Huang et al. 2023, Zhou et al. 2024]. Além disso, a chegada do padrão de telefonia celular 5G e as projeções para o 6G introduzem desafios sem precedentes de escalabilidade, com a necessidade de suportar conectividade massiva, estimada em até 10 milhões de dispositivos por km^2, bem como prover comunicação com latência de submilissegundo [Boateng et al. 2024b]. A integração de múltiplos domínios, como os segmentos satelital, aéreo e terrestre, amplia ainda mais a complexidade operacional e a carga de gerenciamento dessas redes [Zhou et al. 2024].

As abordagens convencionais de Gerenciamento de Redes e Serviços (*Network and System Management* – NSM), baseadas em *scripts*, regras fixas e algoritmos estáticos, têm se mostrado inadequadas para lidar com a volatilidade e a complexidade das redes modernas. Essas soluções carecem de flexibilidade para se adaptar a mudanças na infraestrutura em tempo real e apresentam dificuldades no tratamento de dados não estruturados, como *logs* complexos e intenções expressas em linguagem natural. O processo de configuração manual é frequentemente descrito como trabalhoso, propenso a erros e dispendioso, sendo que uma única falha na configuração de uma lista de controle de acesso (*Access Control List* – ACL) pode causar interrupções graves na rede [Huang et al. 2023, Zhou et al. 2024]. Diante desse cenário, emerge o conceito de NetOps 2.0, também referido como inteligência de rede unificada, que foca na automação, agilidade e análise de dados para substituir processos manuais e estáticos [Huang et al. 2023]. O paradigma de NetOps 2.0 alinha-se ao conceito de *Zero-touch Network & Service Management* (ZSM), no qual a rede busca alcançar capacidades de auto-operação, auto-manutenção e auto-otimização, com intervenção humana mínima ou nula [Lira et al. 2024]. Nesse contexto, consolida-se a chamada "Era da Comunicação por Agentes", em que entidades inteligentes utilizam raciocínio e percepção para orquestrar ferramentas e protocolos de forma autônoma [Kong et al. 2025, Derouiche et al. 2025].

A evolução de Modelos de Linguagem de Grande Escala (do inglês, *Large Language Models* - LLMs), Modelos de Linguagem Pequenos (do inglês, *Small Language Models* - SLMs) e técnicas como Geração Aumentada (do inglês, *Retrieval-Augmented Generation* - RAG) por Recuperação amplia as capacidades de NetOps 2.0, permitindo raciocínio sobre estados de rede, integração com ferramentas externas e execução de ações de forma controlada [Derouiche et al. 2025, Chowa et al. 2026]. Estudos recentes [Long et al. 2025, Boateng et al. 2024b] mostram que esses modelos de IA aplicados a NetOps 2.0 são capazes de raciocinar sobre configurações, documentação técnica e estados complexos de sistemas, além de interagir com ferramentas externas e executar

ações de forma autônoma ou semi-autônoma. Nesse contexto, agentes de IA tornam-se particularmente relevantes para a área de redes de computadores, pois são capazes de interpretar solicitações de alto nível expressas em linguagem natural, consultar bases de conhecimento estruturadas ou não estruturadas, planejar sequências de ações e executar comandos de maneira validada e controlada.

Neste minicurso, serão apresentados os conceitos para projetar e implementar agentes inteligentes para automação e configuração de redes de computadores, com foco em arquiteturas práticas e reprodutíveis. A abordagem adotada no minicurso privilegia o uso de SLMs e LLMs executados localmente, a integração com ferramentas externas e a definição de fluxos de execução seguros, permitindo a experimentação em ambientes controlados antes da generalização para cenários reais mais complexos. Como estudo de caso prático, o documento explora a construção de agentes para automação de redes em ambientes Linux, utilizando *iproute2* e *network namespaces* como uma plataforma controlada e reprodutível para experimentação e compreensão do funcionamento desses agentes. Ao final do minicurso, os participantes estarão capacitados a integrar conceitos de IA e administração de sistemas, projetando agentes aplicáveis a diferentes ambientes e cenários de automação inteligente de redes de computadores.

O restante deste documento está organizado da seguinte forma. A Seção 3.2 apresenta os fundamentos de agentes inteligentes e modelos de linguagem. A Seção 3.3 discute arquiteturas de agentes com SLMs e LLMs. A Seção 3.4 introduz o paradigma de IA Agêntica e seu ciclo deliberativo. A Seção 3.5 aborda a engenharia de *prompts*. A Seção 3.6 explora o uso de RAG para aumento de precisão. A Seção 3.7 descreve a arquitetura prática baseada em Python, Ollama e OpenWebUI. A Seção 3.8 detalha a implementação do agente configurador. A Seção 3.9 discute aspectos de segurança. Por fim, a Seção 3.10 apresenta as conclusões.

3.2. Agentes Inteligentes e Modelos de Linguagem

Diferentemente dos sistemas tradicionais de NSM, baseados em regras estáticas e fluxos pré-definidos, as abordagens convencionais apresentam limitações diante da complexidade, dinamicidade e escala das redes modernas. Nesse contexto, consolida-se o paradigma de NetOps 2.0, no qual a operação da rede passa a incorporar mecanismos capazes de interpretar objetivos de alto nível, tomar decisões com base no contexto e adaptar seu comportamento ao longo do tempo. Essa mudança desloca o foco da configuração manual para a definição de intenções, viabilizando uma operação mais declarativa e orientada a resultados. Destacam-se três capacidades fundamentais: interpretação de intenção (*Intent-Driven*), planejamento automático e diagnóstico inteligente.

A Interpretação de Intenção é responsável por traduzir objetivos de alto nível, expressos por operadores ou por requisitos de negócio, em metas técnicas concretas para a rede [Boateng et al. 2024b]. Essa capacidade reduz a complexidade administrativa ao aproximar a linguagem humana das configurações operacionais, permitindo que a infraestrutura atue de acordo com políticas orientadas a resultados [Lira et al. 2024]. Em vez de o operador precisar especificar manualmente cada comando e cada dependência técnica, torna-se possível expressar a finalidade desejada, cabendo ao sistema interpretar como esse objetivo deve ser materializado.

No planejamento automático, uma vez definida a intenção, a rede deve determinar quais ações precisam ser executadas para satisfazê-la. Esse processo é essencial em cenários que exigem raciocínio multietapa, coordenação de recursos e adaptação contínua, como no fatiamento de rede (*network slicing*), na alocação dinâmica de recursos e na orquestração de serviços sob demanda [Boateng et al. 2024b, Zhou et al. 2024]. Em termos práticos, essa capacidade permite decompor uma solicitação ampla em uma sequência coerente de subtarefas, respeitando dependências técnicas, restrições de segurança e o estado corrente da infraestrutura.

Por fim, no diagnóstico inteligente, após a execução, a rede deve ser capaz de identificar desvios, falhas e degradações de desempenho. Em ambientes distribuídos e de larga escala, cada vez mais comum nos dias atuais, o *troubleshooting* manual tende a ser lento, custoso e sujeito a erro. Nesse contexto, técnicas baseadas em LLMs e agentes inteligentes podem apoiar a análise de *logs*, a correlação de eventos, a interpretação de telemetria e a inferência de causas prováveis, acelerando a detecção e a resolução de problemas [Huang et al. 2023, Boateng et al. 2024b].

Em conjunto, essas capacidades formam um ciclo de operação inteligente para o NetOps 2.0, no qual a rede compreende a intenção, planeja as ações necessárias, executa adaptações e diagnostica eventuais falhas para corrigir seu comportamento. A integração entre IA generativa e agentes autônomos desponta como um caminho promissor para transformar as infraestruturas de rede em sistemas mais resilientes, responsivos e adaptativos [Boateng et al. 2024b].

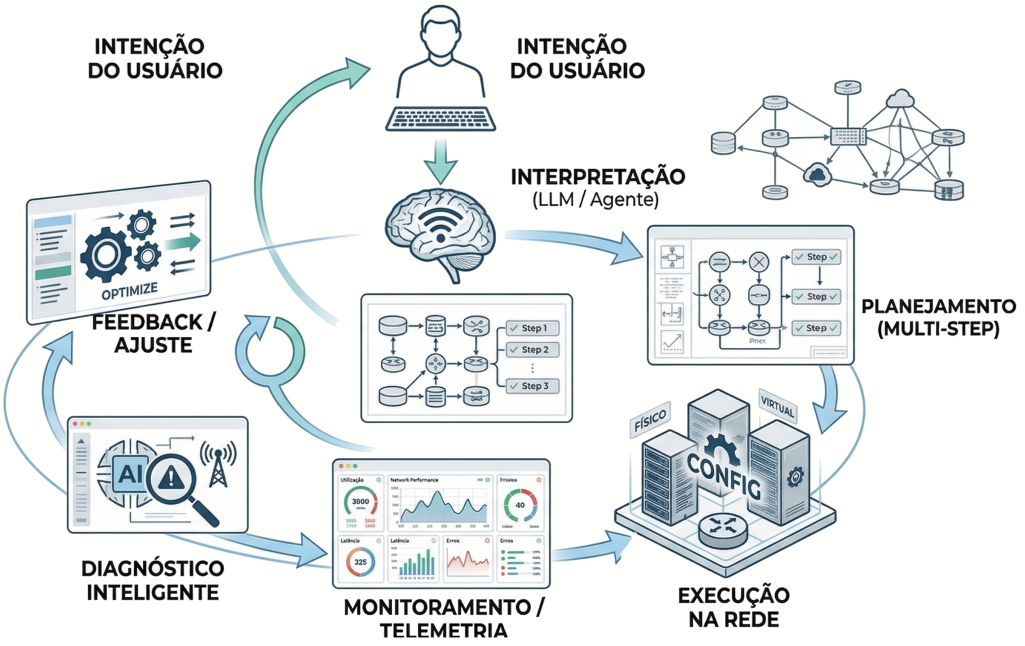


Figura 3.1. Ciclo de processamento de IA em redes

A Figura 3.1 apresenta um exemplo do ciclo de processamento de IA em redes envolvendo desde a intenção do usuário até a execução de operações na rede. Mais do que um conjunto abstrato de ideias, esse ciclo estabelece a base operacional dos sistemas agênticos aplicados à rede. Na prática, ele pode ser implementado como um laço iterativo

entre interpretação, planejamento, execução e validação, no qual o modelo de linguagem atua como núcleo cognitivo, enquanto ferramentas externas executam ações e devolvem observações sobre o estado do ambiente. Essa visão será retomada nas próximas seções, culminando na implementação prática de um agente configurador de redes.

3.2.1. Ecossistema de Modelos de IA Generativa

Em vez de depender de um único modelo monolítico, a arquitetura moderna de sistemas inteligentes distribui responsabilidades entre componentes especializados, buscando modularidade, determinismo, eficiência e manutenibilidade [Bandara et al. 2025]. Essa abordagem permite que diferentes modelos assumam papéis distintos em cada subetapa de uma tarefa complexa, formando um ecossistema cooperativo no qual cada interação faz parte de um fluxo maior de decisão e execução. No contexto de redes de computadores, essa organização é particularmente relevante. Tarefas como interpretar uma solicitação, consultar documentação, planejar mudanças, validar comandos, executar ações e verificar resultados possuem requisitos distintos de latência, precisão, custo e auditabilidade. Por essa razão, sistemas agênticos modernos podem empregar diferentes tipos de modelos especializados, entre os quais se destacam:

- **Modelos Base (*Fast Models*):** são modelos menores ou destilados, otimizados para baixa latência e custo reduzido. Em ambientes de rede, são adequados para tarefas rotineiras, como classificação de intenção, extração de parâmetros de configuração e identificação rápida do tipo de solicitação recebida [OpenAI 2025].
- **Modelos de Raciocínio (*Reasoning Models*):** são voltados a capacidades cognitivas superiores e utilizam estratégias como *Chain-of-Thought* (CoT) ou *Tree-of-Thought* (ToT) para decompor problemas em etapas lógicas. Em redes, são úteis para *troubleshooting* multietapa, planejamento de alterações que envolvem vários dispositivos e análise de dependências entre ações [Yao et al. 2023].
- **Modelos Agentes (*Tool-Using Models*):** atuam como núcleo cognitivo do fluxo de trabalho, sendo treinados ou configurados para decidir quais APIs, comandos ou ferramentas externas devem ser invocados, em que momento isso deve ocorrer e como os resultados devem ser incorporados ao processo decisório [Schick et al. 2023]. No domínio de NSM, isso inclui execução de comandos via SSH, consulta a sistemas de monitoramento, controladores SDN, inventários ou mecanismos de validação.
- **Modelos de *Embeddings*:** são fundamentais para converter documentos textuais, registros operacionais e até artefatos visuais em representações vetoriais, viabilizando busca semântica e recuperação eficiente de conhecimento [Park et al. 2023]. Em redes, permitem localizar trechos relevantes de manuais, *playbooks*, RFCs, incidentes passados e configurações históricas.
- ***Verifier / Critic*:** atuam como auditores de qualidade, verificando as saídas produzidas por outros modelos para identificar alucinações, erros de sintaxe, violações de política ou inconsistências operacionais [Zhou et al. 2024]. Em cenários de rede, podem ser associados a validadores sintáticos, simuladores, políticas internas ou ferramentas como Batfish.

- ***Router / Orchestrator*:** funcionam como coordenadores do fluxo de trabalho, selecionando o modelo ou recurso mais adequado para cada subtarefa [Lira et al. 2024]. Em uma arquitetura de rede, esse componente pode decidir quando usar um SLM local de baixa latência e quando escalar o problema para um LLM mais robusto.
- ***Memory Models*:** gerenciam retenção de informações em diferentes escalas temporais, combinando memória de curto prazo, associada à janela de contexto, com memória de longo prazo, normalmente baseada em armazenamento externo, como bancos vetoriais ou históricos operacionais [Park et al. 2023]. Em NSM, isso permite incorporar topologias, estados anteriores, incidentes passados e decisões já tomadas.

O sistema apresentado neste minicurso materializa essa organização em um cenário de redes, no qual cada componente assume um papel específico no ciclo operacional. Modelos rápidos podem ser usados para classificação de intenção e extração de parâmetros. Modelos de raciocínio podem apoiar a decomposição de tarefas complexas. Modelos agentes podem interagir com ferramentas externas. Finalmente, módulos verificadores podem garantir a conformidade das ações antes da aplicação. Essa arquitetura modular aproxima os conceitos de IA generativa das práticas reais de NSM, além de facilitar a substituição ou evolução de componentes conforme os requisitos de desempenho e segurança [Lira et al. 2024].

As distinções entre as diferentes tecnologias de IA generativa são frequentemente confundidas, embora desempenhem papéis bastante distintos em um ecossistema inteligente. Essa distinção é particularmente importante em redes, nas quais não basta gerar texto tecnicamente plausível. É necessário compreender objetivos, consultar contexto, decidir ações, interagir com ferramentas e validar resultados antes de qualquer aplicação sobre a infraestrutura.

Os **chatbots** são aplicações voltadas à interação conversacional, nas quais o usuário fornece um *prompt* e o sistema retorna uma resposta em um ciclo simples de pergunta e resposta [Bandara et al. 2025]. Sua capacidade de planejamento tende a ser inexistente ou depende inteiramente da condução manual do usuário, já que não decompõem tarefas por conta própria. Sua execução também é limitada, restringindo-se essencialmente à geração de texto. Quando utilizam ferramentas externas, isso ocorre de forma passiva ou bastante restrita, sem orquestração autônoma. No domínio de redes, seu uso é mais apropriado para suporte informacional, como explicar comandos, descrever protocolos ou interpretar mensagens de erro, mas não para operar diretamente a infraestrutura [OpenAI 2025, Kong et al. 2025].

Os **LLMs** são modelos treinados em grandes volumes de dados textuais para compreender e gerar linguagem humana, funcionando como núcleo cognitivo de muitos sistemas de IA [Kong et al. 2025, de Lamo Castrillo et al. 2025]. Embora sua operação fundamental esteja baseada na previsão do próximo *token*, esses modelos apresentam capacidades emergentes de raciocínio e lógica, o que lhes confere potencial para planejamento, por exemplo, por meio de estratégias como *Chain-of-Thought* [Kong et al. 2025, Du et al. 2026]. Em redes, podem gerar configurações, sugerir comandos, resumir logs ou apoiar o diagnóstico de falhas. No entanto, sua atuação isolada continua predominantemente reativa. Mesmo quando integrados a mecanismos de *function*

calling, os LLMs puros normalmente não mantêm controle autônomo contínuo do fluxo operacional, dependendo de um sistema externo para decidir quando executar, validar ou corrigir ações [Schick et al. 2023, Kong et al. 2025].

Por outro lado, os **SLMs** são versões compactas de modelos de linguagem, frequentemente obtidas por técnicas como destilação de conhecimento a partir de modelos maiores [Sanh et al. 2020]. Sua principal vantagem está na eficiência computacional, com menor latência e menor custo, o que favorece sua execução em dispositivos de borda ou ambientes com recursos limitados [Sanh et al. 2020, Zhou et al. 2024]. Quando ajustados para domínios específicos, como NSM, podem apresentar desempenho elevado em tarefas estruturadas, como geração de comandos de configuração, validação de sintaxe, classificação operacional e execução de fluxos padronizados de automação. Em aplicações com uso de ferramentas, SLMs podem alcançar desempenho comparável ou até superior ao de LLMs em *tool calling* e *function calling*, com custo significativamente menor, da ordem de 10 a 100 vezes [NVIDIA Research 2025]. Sua autonomia tende a ser de média a alta em tarefas repetitivas e especializadas, embora permaneça mais restrita fora de domínios estreitos.

Por fim, os **agentes inteligentes** representam uma evolução em que o modelo de IA deixa de atuar apenas como gerador de respostas e passa a desempenhar o papel de uma entidade capaz de agir sobre o ambiente [Kong et al. 2025]. Nesses sistemas, o modelo é utilizado para interpretar objetivos, decompor tarefas, construir planos, invocar ferramentas, monitorar resultados e ajustar suas decisões dinamicamente conforme o contexto. Sua execução é ativa, pois o agente não apenas produz texto, mas realiza ações concretas no ambiente digital ou físico por meio de ferramentas e serviços externos. Em redes, isso significa consultar telemetria, recuperar documentação, gerar comandos, executar ações por SSH, API ou NETCONF, verificar resultados e decidir se a tarefa foi concluída ou se requer correção. Essas características se apoiam em quatro propriedades fundamentais: percepção, memórias de curto e longo prazo, raciocínio ou planejamento e capacidade de ação [Kong et al. 2025, Elkael et al. 2026].

A Tabela 3.1 sintetiza essas diferenças em termos de planejamento, execução, uso de ferramentas e autonomia. Essa distinção é crucial para o entendimento do NetOps 2.0, no qual o objetivo não é apenas disponibilizar uma interface conversacional, mas empregar sistemas capazes de planejar, decidir e executar configurações de rede de forma controlada e progressivamente mais autônoma [Elkael et al. 2026, Lira et al. 2024].

3.2.2. Classificação por Tamanho

Os LLMs são caracterizados por possuírem bilhões ou até trilhões de parâmetros, sendo treinados em escalas massivas de dados para atingir capacidades versáteis de compreensão e raciocínio [Huang et al. 2023]. Modelos icônicos, como o GPT-3, possuem cerca de 175 bilhões de parâmetros, enquanto estimativas não confirmadas atribuem ao GPT-4 uma escala ainda superior [Bommasani et al. 2022]. Por outro lado, os SLMs são versões compactas e otimizadas, muitas vezes derivadas por destilação de modelos maiores, resultando em modelos mais leves e rápidos em tempo de inferência [Sanh et al. 2020, Bommasani et al. 2022]. A geração de modelos a partir de 2025 avançou significativamente além dos primeiros SLMs, com modelos como Phi-4-mini,

Tabela 3.1. Comparação simplificada entre chatbots, LLMs, SLMs e agentes

Categoria	Planejamento	Execução	Ferramentas	Autonomia
Chatbot	Manual / inexistente	Texto	Limitado	Baixa
LLM	Potencial (CoT)	Texto / código	Reativo (*function calling*)	Baixa
SLM	Alta em domínio específico	Rápida / estruturada	Ativo, com baixo custo	Média
Agente	Alta e dinâmica	Ativa, com ações sobre o ambiente	Orquestrado	Alta

LFM2.5, Qwen-2.5-7B e Llama-3.2 demonstrando que o limiar de capacidade necessário para tarefas agênticas estruturadas está abaixo do que se estimava anteriormente [Microsoft Research 2025, Liquid AI 2025, NVIDIA Research 2025]. A Tabela 3.2 apresenta um sumário de modelos SLM relevantes para agentes a partir de 2025.

Tabela 3.2. Modelos SLM relevantes para agentes a partir de 2025

Modelo	Parâmetros	Destaque para Agentes
Microsoft Phi-4-mini	3,8B	Raciocínio, *coding* e *function-calling* [Microsoft Research 2025]
Liquid AI LFM2.5	1,2B	Arquitetura não-*Transformer*; agentes *edge* e raciocínio *on-device* [Liquid AI 2025]
Qwen-2.5-7B	7B	Desempenho balanceado e bom suporte a *tool calling*
Llama-3.2-1B/3B	1–3B	Implantação ultra-leve em *mobile* e IoT
Ministral-3B/8B	3–8B	Especialização em chamadas de ferramentas

A distinção entre essas classes de modelos pode ser analisada a partir de quatro pilares principais. O primeiro é o **número de parâmetros**. Enquanto LLMs de ponta operam na escala de centenas de bilhões ou mais, os SLMs tendem a situar-se abaixo da faixa de 10 bilhões de parâmetros, priorizando eficiência sem abandonar completamente a capacidade de generalização [Boateng et al. 2024b]. O segundo pilar é o **contexto**. LLMs modernos oferecem janelas de contexto muito amplas, permitindo processar grande volume de documentação, histórico operacional e instruções em uma única interação. SLMs tendem a trabalhar com contextos menores, embora versões recentes tenham ampliado essa capacidade [Chowa et al. 2026]. Em redes, isso influencia diretamente o quanto de topologia, política, documentação e estado da sessão pode ser analisado de uma só vez. O terceiro pilar é o **custo**. O treinamento de um LLM de fronteira é proibitivo

para a maior parte das organizações, e sua inferência também tende a ser significativamente mais cara [Zhou et al. 2024]. Em contrapartida, SLMs apresentam custos mais acessíveis e tornam-se especialmente atrativos para tarefas rotineiras, frequentes e bem delimitadas [Zhou et al. 2024, Chowa et al. 2026]. O quarto pilar é a **infraestrutura**. LLMs normalmente exigem grande disponibilidade de GPUs e alto consumo energético para treinamento e inferência, sendo frequentemente inviáveis para execução contínua em ambientes locais ou distribuídos [Zhou et al. 2024]. Já os SLMs são mais adequados para executar em hardware comum, servidores de borda ou até dispositivos com recursos limitados [Sanh et al. 2020].

A escolha entre processar tarefas de IA na nuvem ou localmente envolve um compromisso entre desempenho, latência, privacidade e custo. Modelos em nuvem oferecem acesso direto a LLMs de maior capacidade, com abundância de recursos computacionais para tarefas complexas [Zhou et al. 2024]. No entanto, também apresentam maior latência de ponta a ponta, custos associados ao uso contínuo da API e riscos decorrentes do envio de dados sensíveis da infraestrutura para ambientes externos [Chowa et al. 2026]. Em contraste, modelos locais, seja na borda ou *on-device*, são mais adequados para aplicações que exigem resposta rápida, maior controle sobre os dados e menor dependência de conectividade [Zhou et al. 2024]. Em ambientes de rede, essa diferença é crítica, pois o tempo de resposta e a confidencialidade das informações operacionais podem ser fatores determinantes.

Dessa forma, a tendência mais promissora aponta para arquiteturas híbridas, nas quais a nuvem é utilizada para tarefas de maior complexidade e planejamento de alto nível, enquanto modelos locais, tipicamente menores, são responsáveis pela inferência rotineira e por tarefas específicas de domínio [Chowa et al. 2026, Zhou et al. 2024]. Esse arranjo é particularmente aderente ao gerenciamento de redes, onde a maior parte das operações é repetitiva e previsível, mas uma parcela menor demanda raciocínio mais profundo e maior poder de generalização.

Tese SLM-First e o Novo Paradigma Operacional

A execução local de modelos de larga escala impõe desafios significativos em termos de custo computacional, consumo de energia e requisitos de infraestrutura. Por outro lado, o uso contínuo de modelos em nuvem também envolve custos operacionais relevantes, como cobrança por requisição, latência de comunicação e transferência de dados. Nesse contexto, pesquisas recentes defendem que SLMs na faixa de 1 a 12 bilhões de parâmetros são suficientemente poderosos, mais adequados e economicamente mais viáveis para a maioria das invocações em sistemas agênticos [NVIDIA Research 2025]. Isso ocorre porque grande parte das tarefas executadas por agentes é simples, repetitiva e estruturalmente bem definida, como classificação de intenção, extração estruturada, roteamento, preenchimento de argumentos e chamadas de função. Tais tarefas não exigem, necessariamente, a capacidade de raciocínio aberto de modelos *frontier*. Nesses casos, SLMs ajustados para domínios específicos tendem a gerar saídas mais aderentes a formatos estruturados do que LLMs maiores, reduzindo a necessidade de pós-processamento e o número de *retries*, a um custo 10 a 100 vezes menor [NVIDIA Research 2025].

Nesse cenário, consolidam-se os padrões **SLM-default** e **LLM-fallback** [Microsoft Research 2025]. Nesse modelo, as requisições são inicialmente roteadas para um SLM rápido, frequentemente executado localmente. Em seguida, um módulo verificador avalia a qualidade, a confiança ou a conformidade da saída produzida. Caso o resultado seja insuficiente, a tarefa é escalonada para um LLM mais robusto, normalmente disponível em infraestrutura mais poderosa ou em nuvem. Essa abordagem reduz significativamente custos e latência, sem abrir mão da capacidade de lidar com casos complexos ou fora da distribuição do domínio rotineiro.

As métricas de engenharia associadas ao uso de SLMs em produção refletem objetivos práticos, como custo por tarefa bem-sucedida, taxa de validade de *schema*, taxa de chamadas executáveis, latências p50 e p95[1] e energia por requisição [NVIDIA Research 2025]. Em ambientes de rede, essas métricas são particularmente úteis porque aproximam a avaliação do modelo de critérios operacionais reais, e não apenas de *benchmarks* genéricos de linguagem.

3.2.3. Limitações

Apesar do potencial transformador, a adoção de modelos de linguagem e agentes inteligentes em redes enfrenta barreiras técnicas e operacionais que devem ser tratadas com rigor. A seguir são apresentadas algumas limitações, a saber: alucinações, risco operacional e dependência de contexto.

Alucinações correspondem à geração de informações factualmente incorretas, sem sentido ou fictícias, frequentemente apresentadas com elevado grau de confiança [Zhou et al. 2024, Lira et al. 2024]. Os modelos podem inventar fatos, embelezar conhecimento incompleto ou produzir associações técnicas incorretas com aparência plausível [Park et al. 2023]. Em infraestruturas críticas, como telecomunicações e redes corporativas, esse comportamento compromete diretamente a confiabilidade das soluções produzidas.

Não-determinismo deve-se a natureza probabilística dos LLMs implica que estados idênticos da rede podem produzir ações ligeiramente diferentes em execuções sucessivas [Elkael et al. 2026]. Essa variabilidade afeta a consistência e a previsibilidade dos planos de execução, dificultando seu uso em funções críticas ou fortemente auditadas [de Lamo Castrillo et al. 2025]. Embora parâmetros como temperatura permitam controlar parcialmente a variabilidade, o modelo continua operando sob uma distribuição probabilística, e não sob uma lógica estritamente determinística [Ha and Schmidhuber 2018].

Risco Operacional deve-se ao fato que a integração de LLMs pode introduzir pontos únicos de falha, já que erros, vieses ou limitações do modelo base tendem a se propagar para aplicações derivadas [Bommasani et al. 2022]. Além disso, a opacidade desses sistemas limita sua aceitação em contextos que exigem explicabilidade e previsibilidade [Lira et al. 2024, Boateng et al. 2024b]. O uso de modelos em nuvem também impõe riscos à privacidade e à segurança, uma vez que demanda o compartilhamento de

[1]Latências p50 e p95 são métricas de percentil usadas para medir o tempo de resposta de sistemas (e.g., APIs e bancos de dados). O p50 (mediana) representa o tempo em que 50% das solicitações são concluídas, indicando o cenário típico. O p95 (percentil 95) indica o tempo para 95% das solicitações, mostrando o desempenho dos casos extremos.

dados potencialmente sensíveis da infraestrutura [Lira et al. 2024]. Em cenários extremos, um agente mal alinhado ou comprometido pode produzir efeitos concretos e graves sobre sistemas reais [Kong et al. 2025].

Dependência de Contexto deve-se ao fato que a desempenho dos modelos é limitado pela janela de contexto, que restringe a quantidade de informação processável simultaneamente [de Lamo Castrillo et al. 2025]. Além disso, pequenas variações na redação de *prompts* podem levar a comportamentos distintos, o que torna a engenharia de *prompts* um elemento central na construção de sistemas mais estáveis [Bommasani et al. 2022]. Em redes, isso se torna ainda mais sensível quando a tarefa depende de documentação extensa, estado atualizado do ambiente e múltiplas restrições operacionais.

Essas limitações reforçam a necessidade de mecanismos de verificação automática, supervisão humana e implementação de *guardrails* antes de qualquer aplicação em redes de produção [Elkael et al. 2026].

Capacidades e Riscos Específicos em Tarefas de Rede

As limitações descritas anteriormente assumem formas concretas quando o domínio é a gerência de redes. Do lado das capacidades, os modelos demonstram desempenho consistente em tarefas bem delimitadas, como geração de trechos padronizados de configuração (e.g., interfaces, VLANs, ACLs e prefixos de roteamento) a partir de instruções em linguagem natural; *parsing* e sumarização de logs; tradução de intenção operacional para sintaxe de equipamentos quando exemplos do fabricante estão presentes no contexto; e explicação de mensagens de falha para operadores menos experientes [Chakraborty et al. 2024, Lira et al. 2024].

O risco, contudo, cresce em cenários mais sensíveis. Em topologias inéditas, combinações de equipamentos e protocolos ausentes dos dados de treinamento podem levar o modelo a extrapolar, gerando configurações sintaticamente válidas, porém semanticamente incorretas. Em ambientes multivendedor, a mistura de sintaxes e convenções específicas de fabricantes distintos aumenta a probabilidade de alucinações em campos delicados. Em comandos com efeito imediato e irreversível, como desligamento de interfaces, remoção de rotas ou alterações em ACLs em produção, o custo de uma decisão incorreta pode ser operacionalmente crítico [Boateng et al. 2024b]. Além disso, quando a tarefa exige raciocínio sobre estado global, o modelo corre o risco de decidir com base apenas no contexto textual fornecido, e não no estado real e atualizado da rede.

Essa assimetria define uma estratégia prudente de adoção: agentes podem atuar como primeira linha em tarefas rotineiras e bem delimitadas, mas com revisão humana ou verificação automática obrigatória antes da aplicação de qualquer configuração em produção [Lira et al. 2024]. Em outras palavras, a inteligência do agente deve ser combinada com controle operacional rigoroso.

Instrução em Linguagem Natural à Configuração Aplicada

Para ilustrar de forma integrada os conceitos discutidos, incluindo interpretação de intenção, planejamento, uso de memória, execução e verificação, apresentamos, a seguir, o fluxo completo de operação de um agente inteligente a partir de uma instrução em linguagem natural até a aplicação efetiva de uma configuração na rede.

Considere um operador que instrui o agente:

"Isole a VLAN 200 na interface eth2 do switch core-01 e confirme que o trunk para o switch dist-02 continua operacional."

1. Percepção e decomposição. A instrução chega como texto ao sistema. O modelo identifica duas subtarefas principais: configurar o isolamento da VLAN em uma interface de acesso e verificar o estado do *trunk* para outro equipamento. Esse passo corresponde à interpretação inicial da intenção e à decomposição do objetivo em ações menores.

2. Consulta à memória de longo prazo. O agente invoca mecanismos de recuperação para buscar documentação do fabricante do *core-01*, padrões operacionais internos e *playbooks* relacionados a VLANs. O conteúdo recuperado é inserido no contexto como suporte factual à tomada de decisão.

3. Geração da configuração. Com parâmetros de geração controlados, o modelo produz os comandos na sintaxe apropriada do equipamento. A saída idealmente não é texto livre, mas uma estrutura processável pelo executor, por exemplo em JSON:

```
{
  device: "core-01",
  commands:
  [
    "interface eth2",
    "switchport mode access",
    "switchport access vlan 200"
  ]
}
```

4. Verificação antes da aplicação. Antes da execução, um módulo verificador analisa a configuração produzida de acordo com as políticas definidas e das restrições do ambiente. Uma vez aprovada, a ação é enviada ao dispositivo por meio de mecanismos como SSH, NETCONF ou APIs equivalentes, e os resultados são coletados.

5. Observação e avaliação. O retorno da execução volta ao agente, que avalia se o estado obtido corresponde ao objetivo inicial. Caso o *trunk* para o *dist-02* permaneça operacional, a tarefa é concluída e registrada. Caso contrário, o agente formula uma ação corretiva, ajusta seu plano e itera.

Esse fluxo, que um operador humano poderia executar em vários minutos mediante consultas manuais à documentação, pode ser realizado em segundos, com rastreabilidade completa. Esse é um dos ganhos práticos mais imediatos da arquitetura de agentes aplicada a redes [Lira et al. 2024, Chakraborty et al. 2024].

Dessa forma, a combinação entre modelos de linguagem, memória, ferramentas e mecanismos de validação estabelece a base para a construção de sistemas capazes de atuar diretamente sobre a infraestrutura de rede. No entanto, a materialização desses conceitos exige uma arquitetura prática que integre decisão, execução e controle operacional. Essa transição entre fundamento conceitual e implementação será aprofundada nas seções seguintes.

3.3. Arquiteturas de Agentes com SLMs e LLMs

A construção de agentes inteligentes para redes pode seguir diferentes arquiteturas, cuja escolha depende diretamente de requisitos como latência, privacidade, custo operacional, capacidade de raciocínio e grau de autonomia desejado. Em ambientes de rede, essa decisão é particularmente sensível, pois o sistema precisa equilibrar precisão técnica, controle sobre dados sensíveis e tempo de resposta compatível com a operação. Não existe, portanto, uma arquitetura única e universalmente superior. Em vez disso, diferentes arranjos podem ser mais adequados conforme o contexto de implantação e o tipo de tarefa a ser executada.

3.3.1. Possíveis Abordagens

Uma primeira abordagem consiste no uso de um **LLM externo acoplado a uma aplicação local**, normalmente implementada em Python, que atua como orquestradora. Nessa arquitetura, a aplicação local recebe a solicitação do operador, organiza o contexto, classifica a intenção, monta o *prompt* e envia a requisição a um modelo robusto hospedado externamente por meio de API [Lira et al. 2024, Kong et al. 2025]. O sistema LLM-NetCFG é um exemplo representativo dessa estratégia, pois utiliza um módulo intermediário para gerenciar os *prompts*, estruturar entradas e apoiar a classificação de intenções antes do envio ao modelo [Lira et al. 2024]. A principal vantagem dessa abordagem está no acesso a modelos de alta capacidade sem a necessidade de manter infraestrutura local de grande porte. Em contrapartida, há dependência de conectividade, maior latência de ponta a ponta e exposição potencial de dados sensíveis da rede.

Uma segunda arquitetura é o **tool-calling interno**, na qual o modelo é treinado ou configurado para emitir chamadas de ferramentas em formatos estruturados, como JSON, em vez de apenas produzir texto livre [Kong et al. 2025, de Lamo Castrillo et al. 2025]. Nessa abordagem, o modelo decide quais funções, APIs ou mecanismos de consulta devem ser invocados, seja para obter dados em tempo real, seja para executar ações sobre a infraestrutura [de Lamo Castrillo et al. 2025, OpenAI 2025]. Em redes, isso inclui, por exemplo, consultar telemetria, recuperar documentação, acionar validadores, interagir com controladores SDN ou gerar comandos para execução controlada. Essa arquitetura aumenta a capacidade operacional do agente, pois aproxima a geração de linguagem da ação efetiva sobre o ambiente.

Uma terceira possibilidade é o **SLM embarcado localmente**, que consiste na execução de modelos compactos diretamente em hardware local, servidores privados ou dispositivos de borda. Modelos como Phi-4-mini, LFM2.5 e Llama-3.2-3B ilustram essa tendência [Zhou et al. 2024, Microsoft Research 2025, Liquid AI 2025]. Essa arquitetura é especialmente adequada para cenários em que baixa latência, confidencialidade e in-

dependência de conectividade são requisitos centrais [Zhou et al. 2024, Lira et al. 2024]. Em ambientes de redes corporativas, industriais ou de telecomunicações, a execução local reduz a exposição de informações operacionais e permite respostas mais rápidas em tarefas rotineiras, como geração de configurações, classificação de incidentes e validação estrutural de comandos.

Por fim, destacam-se as **arquiteturas híbridas**, nas quais diferentes modelos coexistem em um mesmo *pipeline*, com papéis distintos e complementares. Nesses casos, um componente roteador decompõe tarefas complexas e as distribui entre modelos especializados [Kong et al. 2025]. Modelos com maior capacidade de raciocínio podem ser utilizados para planejamento e análise de casos complexos, enquanto modelos menores e mais rápidos executam subtarefas específicas, como preenchimento de argumentos, chamadas de ferramentas e geração estruturada de saídas [Bandara et al. 2025, OpenAI 2025]. Essa abordagem permite equilibrar custo, latência e qualidade de raciocínio, tornando-se particularmente adequada para ambientes de rede nos quais coexistem tarefas simples e frequentes com situações raras de alta complexidade.

3.3.2. Arquiteturas Avançadas de Raciocínio e Eficiência Energética

Além da escolha da arquitetura geral do sistema, o desempenho de um agente depende também do padrão de raciocínio adotado para organizar o ciclo de decisão e ação. O *framework* ReAct [Yao et al. 2023] estabeleceu um paradigma influente ao propor a sequência Observação $\rightarrow$ Pensamento $\rightarrow$ Ação, permitindo que o modelo intercale raciocínio textual com interação sobre o ambiente. Essa formulação foi importante por tornar o processo mais interpretável e adaptativo. No entanto, em cenários de configuração de redes, nos quais uma sequência incompleta de passos pode produzir falhas em cascata, arquiteturas mais recentes vêm buscando superar suas limitações.

Entre essas abordagens, o **Pre-Act** [Rawat et al. 2025] introduz planejamento explícito antes da execução. Em vez de decidir incrementalmente a cada interação, o agente constrói inicialmente um plano mais completo e detalhado, refinando-o ao longo da execução. Essa característica é particularmente relevante em redes, onde a omissão de um passo intermediário pode comprometer a aplicação correta de VLANs, ACLs, rotas ou políticas distribuídas. Em avaliações, Pre-Act superou ReAct em 70% na métrica *Action Recall*, resultado importante para cenários em que a completude da sequência operacional é crítica. Outra proposta relevante é o **Structured Cognitive Loop (SCL/R-CCAM)** [Kim 2026], que organiza a cognição em cinco fases modulares: *Retrieval*, *Cognition*, *Control*, *Action* e *Memory*. Sua principal contribuição está na introdução de uma camada explícita de controle simbólico, aumentando explicabilidade e aderência a políticas. Em ambientes de produção, essa característica é especialmente valiosa, pois a rede exige não apenas decisões corretas, mas decisões auditáveis e compatíveis com restrições operacionais rígidas.

Já o **Modular Agentic Planner (MAP)** [Webb et al. 2025] propõe decompor o planejamento em módulos especializados, inspirados no funcionamento do córtex pré-frontal humano. Nesse arranjo, diferentes módulos assumem tarefas como monitoramento de erros, proposição de ação, predição de estados e avaliação de resultados. Em estudos comparativos, o MAP superou abordagens como *Chain-of-Thought*, *Multi-Agent Debate*

e *Tree-of-Thought* em diferentes domínios de planejamento. Em redes, esse tipo de modularidade é promissor porque aproxima o raciocínio do agente da própria natureza distribuída e multifásica da operação de infraestrutura. A Tabela 3.3 apresenta uma comparação de arquiteturas avançadas de raciocínio de agentes.

Tabela 3.3. Comparação de arquiteturas avançadas de raciocínio para agentes

Arquitetura	**Abordagem Principal**	**Vantagem-Chave**
ReAct [Yao et al. 2023]	Thought → Action → Observation	Interpretabilidade e adaptabilidade
Pre-Act [Rawat et al. 2025]	Plano completo antes da execução	+70% Action Recall vs ReAct
SCL (R-CCAM) [Kim 2026]	5 fases modulares com controle simbólico	Zero violações de política
MAP [Webb et al. 2025]	Módulos inspirados no PFC humano	Melhor generalização entre tarefas

A dimensão energética também merece atenção especial em sistemas de agentes aplicados a redes. Medições da plataforma ML.ENERGY [ML.ENERGY 2024] indicam que modelos de grande porte, como o LLaMA-65B, podem consumir aproximadamente 3 a 4 J/token, enquanto SLMs operam na faixa de 0,6 mJ/token, uma diferença de até 5.800 vezes. Em experimentos com robótica móvel, observou-se que SLMs foram capazes de restaurar a latência do loop sensório-motor a níveis aceitáveis em tempo real, enquanto LLMs maiores introduziram atrasos proibitivos. Em redes, especialmente em cenários com restrições de latência sub-segundo, esse aspecto deixa de ser apenas econômico e passa a ser operacionalmente decisivo.

3.3.3. Adaptação de SLMs para Cenários de Agentes: *Fine-Tuning* e Quantização

A viabilidade dos SLMs em *pipelines* agênticos depende menos do número absoluto de parâmetros e mais da qualidade do processo de adaptação ao domínio. Modelos na faixa de 1 a 7 bilhões de parâmetros, quando ajustados sobre tarefas de *tool-calling*, geração estruturada ou configuração de redes, frequentemente superam modelos ordens de grandeza maiores que operam apenas via *prompting* genérico [Jhandi et al. 2026, Sharma and Mehta 2025]. Esse resultado é particularmente importante para o domínio de redes, no qual grande parte das tarefas é recorrente, estruturada e dependente de formatos específicos de saída.

O modelo OPT-350M, ajustado sobre o conjunto ToolBench, pode atingir 77,55% de taxa de sucesso em *benchmarks* de chamada de ferramentas, contra 26% do ChatGPT com *chain-of-thought* [Jhandi et al. 2026]. Nesse caso, o treinamento completo levou menos de 15 minutos em hardware de consumidor, indicando que, em tarefas delimitadas, a especialização de um modelo pequeno pode ser mais eficaz do que o uso indiscriminado de modelos muito maiores. A técnica dominante nesse processo é o **QLoRA** [Dettmers et al. 2023], que combina um modelo base quantizado em 4 bits, no formato NF4, com adaptadores de baixo ranque mantidos em precisão plena. Essa estratégia re-

duz drasticamente o consumo de memória, sem perda estatisticamente significativa de desempenho. Na fase de implantação, essa eficiência pode ser ampliada por formatos de quantização pós-treinamento. O **GPTQ** utiliza informação de segunda ordem para minimizar o erro de reconstrução dos pesos em GPU, enquanto o **GGUF**, formato nativo do *llama.cpp* e do Ollama, viabiliza inferência eficiente em CPU, com consumo de memória compatível com servidores de borda. Em avaliações com Llama 3.2-1B submetido a QLoRA seguido de quantização GPTQ 4 bits, a acurácia foi preservada em 0,99 em relação ao modelo de referência em precisão plena [Sharma and Mehta 2025]. Em termos práticos, isso sugere que um SLM executado localmente em um controlador de rede pode desempenhar tarefas de geração de configuração com qualidade comparável à de modelos muito maiores, desde que o domínio de ajuste seja aderente ao ambiente operacional.

O projeto **TinyAgent** [Erdogan et al. 2024] reforça essa hipótese ao mostrar que modelos de 1,1B e 7B parâmetros, ajustados com mecanismos de recuperação de ferramentas para compressão de *prompt*, puderam igualar ou superar o GPT-4-Turbo em benchmarks de chamada de funções, com execução inteiramente local. Para redes industriais, infraestruturas críticas e ambientes com exigências rígidas de confidencialidade, essa característica é particularmente relevante, pois elimina a necessidade de expor topologias, configurações e políticas de roteamento a APIs externas.

Componentes Arquiteturais

A construção de um agente robusto exige a integração coordenada de diferentes módulos funcionais. Em vez de tratar o agente como um único modelo, arquiteturas modernas o definem como um sistema composto por elementos especializados, cada um responsável por uma parte do ciclo operacional.

O **Agent Model** constitui o núcleo cognitivo do sistema, podendo ser um LLM ou um SLM. Esse componente é responsável pela compreensão da linguagem, pelo raciocínio lógico, pela interpretação da intenção e pela formulação do plano de ação [Kong et al. 2025, de Lamo Castrillo et al. 2025].

O **Tool Executor** converte as decisões abstratas do modelo em ações concretas sobre a infraestrutura, como execução de comandos, chamadas de API ou interação com dispositivos e serviços externos [Kong et al. 2025, de Lamo Castrillo et al. 2025]. Em redes, esse componente é central porque estabelece a ponte entre geração semântica e operação técnica real.

O **Verifier** atua como um auditor de segurança, sintaxe e conformidade, validando as configurações geradas antes da aplicação na rede. Esse papel pode ser exercido por regras estáticas, validadores específicos ou ferramentas como Batfish, no caso de análise de políticas e ACLs [Lira et al. 2024].

O **Router** coordena o fluxo entre modelos, agentes ou ferramentas, decidindo qual recurso deve ser utilizado em cada subtarefa [Kong et al. 2025, OpenAI 2025]. Em arquiteturas híbridas, ele é o elemento que viabiliza o padrão SLM-default, LLM-fallback.

A **Memory** é essencial para manter continuidade operacional. Em geral, combina memória de curto prazo, associada ao contexto imediato da interação, com memória de longo prazo, baseada em histórico de interações, estados anteriores, documentos e deci-

sões passadas [Kong et al. 2025].

O **RAG (*Retrieval-Augmented Generation*)** amplia a base factual do agente ao permitir consulta a documentos externos, como manuais técnicos, logs, playbooks e históricos operacionais, reduzindo alucinações e aumentando a precisão da resposta [Kong et al. 2025, de Lamo Castrillo et al. 2025].

Por fim, os mecanismos de **Logging e Auditoria** registram comunicações, decisões e ações executadas, garantindo rastreabilidade, explicabilidade e possibilidade de revisão posterior, características indispensáveis em ambientes críticos [Kong et al. 2025].

3.3.3.1. Arquiteturas Heterogêneas e Roteamento entre SLMs e LLMs

A oposição rígida entre SLMs e LLMs vem perdendo sentido nas arquiteturas de produção mais recentes. Em vez de escolher um único modelo para todo o sistema, arquiteturas heterogêneas distribuem diferentes funções entre modelos distintos, com um componente de roteamento responsável por alocar cada subtarefa ao recurso mais eficiente para aquele contexto [Ye et al. 2025, Yue et al. 2025]. Essa mudança reflete uma compreensão mais realista dos sistemas agênticos: tarefas diferentes impõem exigências diferentes, e raramente um único modelo atende de forma ótima a toda a cadeia de processamento.

O benchmark X-MAS [Ye et al. 2025], que avaliou 27 modelos em cinco domínios por meio de 1,7 milhão de execuções, mostrou que configurações heterogêneas podem superar arquiteturas homogêneas em até 8,4% em raciocínio matemático e em até 47% nos problemas mais complexos da competição AIME. Embora esses números não sejam específicos de redes, eles reforçam o princípio de que a seleção ótima de modelo depende da natureza da subtarefa, e não apenas de uma métrica global de capacidade.

O **MasRouter** [Yue et al. 2025] operacionaliza esse princípio ao adotar uma rede controladora em cascata que decide o modo de colaboração, aloca papéis e roteia cada requisição ao modelo mais apropriado. Em benchmarks de geração de código, o sistema obteve melhora de 1,8 vezes sobre o estado da arte, com redução de custo de 52 vezes em relação ao uso exclusivo de LLM no HumanEval. Dessa lógica emerge o padrão *SLM-default, LLM-fallback*, no qual tarefas rotineiras são encaminhadas a SLMs locais e apenas casos ambíguos, raros ou semanticamente incertos são escalados a LLMs mais robustos [NVIDIA Research 2025]. A Tabela 3.4 sumariza diferentes padrões de implantação de modelos em *pipelines* agênticos para gerência de redes.

No contexto de gerenciamento de redes, esse padrão é bastante aderente à natureza dual das operações. A maior parte das tarefas é repetitiva e bem delimitada, como geração de configurações, verificação de sintaxe, validação de ACLs e classificação de incidentes. Apenas uma fração menor exige raciocínio sobre cenários inéditos, múltiplos sintomas distribuídos ou eventos fora da distribuição de treinamento [Long et al. 2025]. Usar LLMs em todas as etapas gera sobredimensionamento computacional e financeiro; usar apenas SLMs pode limitar a capacidade de resposta a exceções complexas. O roteamento adaptativo resolve essa tensão de forma sistemática [Sharma and Mehta 2025, Kong et al. 2025].

Tabela 3.4. Padrões de implantação de modelos em *pipelines* agênticos para gerência de redes

Padrão	Lógica de roteamento	Caso de uso em redes	Custo relativo
SLM exclusivo	Todas as tarefas no modelo local	Configs rotineiras, baixa latência exigida	Muito baixo
LLM exclusivo	Todas as tarefas via API externa	Diagnóstico complexo sem restrição de custo	Alto
SLM-default/ LLM-fallback	SLM para tarefas conhecidas; LLM para anomalias	Operações diárias com escalonamento pontual	Baixo a médio
Heterogêneo com router	Roteamento dinâmico por complexidade e domínio	Sistemas multiagente autônomos de grande escala	Variável

3.3.4. Comparação com Automação Tradicional

Em relação às abordagens clássicas de automação, os agentes baseados em IA introduzem uma camada deliberativa ausente em ferramentas puramente declarativas. Soluções como Ansible, NETCONF e gNMI continuam fundamentais, mas operam a partir de entradas estruturadas, esquemas previamente definidos e fluxos explicitamente modelados [Huang et al. 2023, Boateng et al. 2024b]. Essa característica lhes confere determinismo e auditabilidade, mas também reduz sua flexibilidade diante de cenários imprevistos, dados não estruturados ou objetivos expressos em linguagem natural.

Ferramentas como o **Ansible** executam scripts e playbooks predefinidos, sendo extremamente úteis em tarefas repetitivas e controladas. No entanto, sua lógica não contempla, por si só, interpretação de intenção, replanejamento dinâmico ou diagnóstico semântico de falhas. Já um **agente deliberativo** utiliza um modelo de linguagem para interpretar a solicitação do operador, planejar as etapas de execução e adaptar seu comportamento com base no retorno do ambiente [Lira et al. 2024, Elkael et al. 2026]. Isso permite, por exemplo, correlacionar mensagens de erro, ajustar um plano intermediário e até corrigir trechos de automação tradicional que tenham falhado [Boateng et al. 2024b].

A comparação torna-se ainda mais clara quando analisamos protocolos específicos. O **NETCONF**, definido pela RFC 6241, oferece um protocolo de gerenciamento baseado em XML sobre SSH, com operações transacionais como `edit-config` e `get-config`, o que garante atomicidade e capacidade de rollback. Sua principal força está na consistência operacional em ambientes multivendedor. Sua limitação, contudo, está na rigidez: o operador precisa conhecer previamente o esquema YANG e a estrutura dos dados envolvidos.

O **gNMI**, por sua vez, adota transporte gRPC e oferece suporte nativo a telemetria em tempo real, tornando-se especialmente útil em ambientes modernos de coleta frequente e observabilidade contínua. Ainda assim, o operador continua precisando conhecer os caminhos exatos dos dados que deseja consultar ou modificar.

O agente inteligente não substitui essas ferramentas, mas opera em uma ca-

mada superior. Ele interpreta a intenção em linguagem natural, decide quais operações NETCONF ou gNMI são necessárias, gera os payloads apropriados, envia as requisições e analisa os resultados devolvidos pelo dispositivo. Quando há falha, o agente pode reformular a ação sem intervenção humana imediata. Em outras palavras, a automação tradicional fornece a interface determinística e auditável; o agente fornece a inteligência adaptativa que a operacionaliza [Huang et al. 2023, Lira et al. 2024].

3.3.5. Infraestrutura de Execução

A viabilidade operacional de agentes inteligentes depende também da infraestrutura utilizada para inferência e execução. A escolha do hardware afeta latência, custo, consumo energético e escalabilidade do sistema.

Em termos gerais, tarefas como recuperação de informação, indexação e pré-processamento podem ser executadas adequadamente em CPU. Já a inferência de modelos maiores tende a exigir GPUs de alto desempenho para atingir tempos de resposta compatíveis com uso interativo ou automação em tempo quase real [Khattab and Zaharia 2020]. Esse aspecto é particularmente relevante em sistemas de agentes, nos quais uma única tarefa pode envolver múltiplas iterações entre modelo, executor e verificador.

LLMs de grande porte apresentam maior custo por token, maior latência e maior demanda computacional. Em contraste, SLMs ajustados e quantizados, especialmente quando combinados com técnicas como LoRA e GGUF, podem oferecer maior vazão de tokens e menor custo operacional [Huang et al. 2023]. Em redes móveis avançadas, infraestruturas 6G ou aplicações críticas com requisitos estritos de tempo de resposta, a latência imposta por modelos remotos ou excessivamente grandes pode tornar-se inviável [Zhou et al. 2024].

Em termos práticos, a escolha da plataforma precisa considerar não apenas desempenho bruto, mas adequação ao contexto da rede. Para tarefas rotineiras, controladas e frequentes, a execução local em CPU ou GPU modesta pode ser suficiente. Para diagnósticos mais complexos ou cenários com forte exigência de raciocínio, pode ser necessário recorrer a infraestrutura mais robusta. A Tabela 3.5 faz uma comparação de plataformas para inferências de modelos.

Tabela 3.5. Comparação de plataformas para inferência de modelos

Plataforma	Tokens/s	Latência	Custo	Maturidade
CPU	5–25	Alta	Baixo	Alta
NVIDIA CUDA	40–300+	Baixa	Alto	Muito Alta
AMD ROCm	60–200	Baixa	Médio	Média
Apple Metal	30–120	Baixa	Médio	Alta (ecossistema Apple)

3.3.6. Classificação por Ambiente

Além do *hardware*, a implantação dos modelos pode ocorrer em diferentes camadas da infraestrutura, cada uma com vantagens e limitações próprias.

No modelo **Cloud**, a inferência é realizada em provedores externos, com acesso a recursos computacionais praticamente ilimitados para treinamento e raciocínio pesado.

Essa abordagem facilita o uso de modelos de ponta, mas aumenta latência e amplia riscos associados à privacidade dos dados [Lira et al. 2024].

Na modalidade **Local**, os modelos são executados dentro da própria rede privada do operador. Isso garante maior controle sobre segurança e confidencialidade, além de reduzir a dependência de conectividade externa [Lira et al. 2024]. Em redes corporativas e ambientes críticos, essa costuma ser uma opção particularmente atraente.

Já na implantação em **Edge**, o modelo é posicionado o mais próximo possível do ponto de operação, como servidores de borda ou estações rádio base. Essa estratégia é indicada para tarefas sensíveis à latência e a decisões em tempo real, embora seja limitada pela capacidade de hardware disponível no dispositivo [Zhou et al. 2024].

Dessa forma, a classificação por ambiente complementa a escolha arquitetural e a escolha de hardware. Em agentes para redes, a decisão final raramente depende de um único fator. Ela resulta da combinação entre requisitos de privacidade, tempo de resposta, custo, criticidade operacional e complexidade do raciocínio esperado.

Como consequência, as arquiteturas mais promissoras para o domínio de redes tendem a combinar execução local ou de borda para tarefas frequentes e sensíveis, com escalonamento seletivo para modelos mais robustos em nuvem quando a complexidade da tarefa justificar esse custo adicional. Essa visão prepara o terreno para a próxima seção, na qual o foco passa do desenho arquitetural para o paradigma operacional da IA Agentic.

3.4. IA Agêntica: Paradigma Moderno para a Construção de Agentes Inteligentes

A IA Agêntica representa uma transição fundamental dos chatbots reativos para sistemas autônomos capazes de raciocinar, planejar e executar tarefas em múltiplas etapas em infraestruturas complexas [Bandara et al. 2025]. Diferentemente do uso tradicional de LLMs por meio de *prompts* únicos, os fluxos agênticos utilizam o modelo de linguagem como um núcleo cognitivo que orquestra ferramentas, gerencia memória e interage dinamicamente com o ambiente [Du et al. 2026, Bandara et al. 2025].

Ciclo deliberativo

O modelo conceitual que orienta a construção desses agentes baseia-se em um ciclo fechado de percepção e ação, permitindo que a IA não apenas gere texto, mas atue como um colaborador inteligente [de Lamo Castrillo et al. 2025].

Perceber (*Perceive*): o agente inicia sua interação capturando dados do ambiente [de Lamo Castrillo et al. 2025]. Em redes, isso inclui o monitoramento de condições de tráfego, logs e indicadores-chave de desempenho (KPIs) [Elkael et al. 2026]. O sistema de percepção transforma esses perceptos brutos em representações significativas que o LLM pode processar, como o estado atual de uma topologia ou uma falha de conectividade detectada [de Lamo Castrillo et al. 2025].

Planejar (*Plan*): com base na percepção e na intenção do operador, por exemplo, "priorizar tráfego de vídeo na Região A", o agente utiliza sua capacidade de raciocínio para decompor o objetivo de alto nível em subtarefas gerenciáveis [Elkael et al. 2026,

de Lamo Castrillo et al. 2025]. O planejamento pode ser hierárquico, distribuindo a execução entre diferentes escalas de tempo, de milissegundos a minutos, e entre diferentes domínios da rede, como núcleo, borda ou RAN [Elkael et al. 2026].

Agir (*Act*): nessa fase, o agente traduz decisões abstratas em operações concretas no mundo digital [de Lamo Castrillo et al. 2025]. O módulo de ação interage com o ambiente por meio de ferramentas externas, como chamadas de API, execução de scripts de configuração ou ajustes de parâmetros em controladores SDN [de Lamo Castrillo et al. 2025, Kong et al. 2025].

Avaliar (*Evaluate/Reflect*): após a execução, o agente entra em um estágio de reflexão e autoavaliação [de Lamo Castrillo et al. 2025]. Nessa etapa, ele analisa os resultados obtidos em relação à intenção original, no contexto de *intent assurance*, para identificar erros de sintaxe ou desvios de configuração, como *intent drift* [de Lamo Castrillo et al. 2025, Elkael et al. 2026]. Se detectar falhas, o agente pode corrigir proativamente seu plano e tentar uma nova abordagem sem intervenção humana imediata [de Lamo Castrillo et al. 2025, OpenAI 2025].

3.4.1. Capacidades Modernas

Os agentes inteligentes modernos superam o processamento estático de texto ao incorporar múltiplas capacidades complementares:

Integração com ferramentas: uso de calculadoras, analisadores de tráfego, simuladores e APIs para realizar operações que excedem as capacidades nativas do LLM [Kong et al. 2025].

Bases de conhecimento via RAG: a *Retrieval-Augmented Generation* (RAG) permite que o agente consulte manuais técnicos, históricos de rede e bases documentais em tempo real, reduzindo alucinações e aumentando a fundamentação factual das respostas [Kong et al. 2025].

Memória de curto e longo prazo: o agente preserva o histórico da sessão para manter coerência interacional e também retém experiências passadas, possibilitando aprendizado a partir de erros e decisões anteriores [Derouiche et al. 2025].

3.4.2. Ecossistema de *Frameworks*

O desenvolvimento desses agentes é facilitado por *frameworks* que padronizam a orquestração, a memória e a comunicação entre componentes:

LangChain Agents: oferece abstrações modulares para planejamento, memória e invocação de ferramentas [Du et al. 2026, Kong et al. 2025].

AutoGen: enfatiza a colaboração multiagente, permitindo que diferentes perfis de IA, como planejadores e verificadores, interajam para resolver problemas complexos [Du et al. 2026, Derouiche et al. 2025].

CrewAI: especializado em processos de trabalho orientados por papéis, permitindo a criação de equipes de agentes com responsabilidades específicas [Derouiche et al. 2025].

OpenAI Swarm e OpenWebUI Tools: representam abordagens emergentes vol-

tadas à orquestração leve de agentes e à integração de ferramentas em interfaces modernas.

Esse ciclo deliberativo transforma o gerenciamento de redes em um sistema de auto-operação, no qual a inteligência evolui continuamente a partir dos dados operacionais e das interações com o ambiente [Elkael et al. 2026].

Comparativo de *frameworks*, protocolos de interoperabilidade e *spec-driven development*

A escolha do *framework* influencia diretamente as características operacionais de todo o sistema de agentes, como indicado na Tabela 3.6.

Tabela 3.6. Comparativo de frameworks de agentes (2025–2026)

Framework	Força principal	Melhor aplicação
LangGraph [LangChain 2025]	*Workflows* com estado e grafos cíclicos	*Workflows* complexos com múltiplos pontos de decisão
CrewAI	Coordenação multiagente orientada por papéis	Prototipagem rápida de sistemas multiagente
AutoGen	Arquitetura orientada a eventos entre agentes	Pesquisa e *workflows* com supervisão humana
LlamaIndex	Indexação e recuperação avançada	Aplicações com forte dependência de RAG
Semantic Kernel	Integração entre LLMs e linguagens corporativas	Ambientes Microsoft e Azure

Model Context Protocol (MCP) [Anthropic 2025]: o MCP consolidou-se como um padrão de fato para integração entre agentes e ferramentas. Baseia-se em mensagens JSON-RPC 2.0 para estabelecer comunicação entre três entidades: *hosts*, que iniciam as conexões; *clients*, que funcionam como conectores internos; e *servers*, que disponibilizam contexto e capacidades. A evolução do protocolo ampliou sua aplicabilidade, inclusive em padrões confiáveis do tipo *call-now/fetch-later*. Entre seus riscos específicos, destacam-se injeção de *prompt* por descrições de ferramentas e escalada de privilégio entre sistemas [Errico et al. 2025].

Agent-to-Agent Protocol (A2A) [Google 2025]: o A2A foi concebido para permitir que agentes implementados em diferentes *frameworks* descubram uns aos outros, autentiquem-se e interajam independentemente da tecnologia subjacente. Seus componentes incluem *Agent Cards*, para descoberta padronizada de capacidades, *Task Management*, para controle do ciclo de vida das tarefas, e mecanismos de segurança corporativa, como OAuth 2.0, chaves de API e mTLS. MCP e A2A são complementares: o primeiro conecta agentes a ferramentas; o segundo organiza a comunicação entre agentes.

***Spec-Driven Development* (SDD)**: quando o custo marginal de geração de código se aproxima de zero, o principal ativo cognitivo deixa de ser o código em si e passa a ser a especificação formal. No SDD, o agente recebe uma especificação estruturada, como

um *schema* JSON, contrato OpenAPI ou grafo de estados, e gera a implementação como artefato derivado, iterando até que verificadores automáticos atestem conformidade. Em redes, isso significa que a intenção do operador, expressa formalmente, torna-se o documento canônico, enquanto a configuração do dispositivo passa a ser apenas sua projeção executável.

3.4.3. *Feedback Loop* Iterativo

A transição de modelos de linguagem para agentes inteligentes ocorre por meio de um sistema de malha fechada (*closed-loop system*), que estabelece uma cadeia contínua de percepção, decisão, ação e realimentação [Kong et al. 2025]. Diferentemente do paradigma de *prompt* único, o agente opera em um ciclo deliberativo que lhe permite executar tarefas complexas e corrigir erros de forma proativa. O fluxo desse ciclo pode ser descrito em quatro etapas principais:

1. **O LLM gera o comando**: o modelo atua como núcleo cognitivo do sistema [de Lamo Castrillo et al. 2025]. Utilizando estratégias como ReAct, intercala raciocínio verbal com a produção de ações específicas [Yao et al. 2023]. Nessa etapa, decide qual ferramenta ou API deve ser invocada e emite comandos estruturados, geralmente em JSON, descrevendo os parâmetros da ação [Kong et al. 2025].
2. **O executor realiza a ação**: as decisões abstratas produzidas pelo LLM são encaminhadas a um módulo executor ou orquestrador. Esse componente traduz o comando em operações reais, como chamadas de API, execução de scripts ou consultas a bases de dados [Lira et al. 2024, de Lamo Castrillo et al. 2025]. Em sistemas como o LLM-NetCFG, o orquestrador garante que a configuração seja enviada corretamente aos dispositivos ou simuladores [Lira et al. 2024].
3. **O resultado retorna ao agente**: após a execução, o ambiente ou a ferramenta externa devolve um resultado, correspondente à fase de observação no paradigma ReAct [Yao et al. 2023]. Esse retorno pode assumir a forma de sucesso operacional, dados de telemetria ou mensagens de erro. No Toolformer, por exemplo, o modelo aprende a utilizar ferramentas avaliando se a resposta obtida é útil para reduzir a incerteza da tarefa [Schick et al. 2023].
4. **O modelo decide continuar ou finalizar**: com base no retorno recebido, o agente reavalia o estado da tarefa. Por meio de mecanismos de autorrefinamento, como o *Self-Refine*, o modelo analisa se o objetivo foi alcançado ou se é necessário ajustar a estratégia [Madaan et al. 2023, Du et al. 2026]. O ciclo prossegue até que uma condição de saída seja satisfeita, como conclusão da tarefa, ocorrência de erro fatal ou atingimento do número máximo de tentativas [OpenAI 2025].

Esse *loop* iterativo é central para o NetOps 2.0, pois permite que o sistema trate falhas de configuração de maneira autônoma. Se um verificador detectar inconsistências, o agente recebe esse retorno e produz uma nova configuração corrigida no ciclo seguinte [Lira et al. 2024].

3.4.4. Planejamento Multietapas

O planejamento multietapas é uma das características que distinguem um agente inteligente de um modelo de linguagem estático. Ele permite decompor objetivos complexos em sequências de tarefas menores e manejáveis, utilizando o LLM como núcleo cognitivo para orquestrar a execução e lidar com incertezas [Zhou et al. 2024, de Lamo Castrillo et al. 2025].

Cadeias de decisão: agentes modernos empregam estratégias de decomposição para navegar em problemas complexos. Essa decomposição pode ocorrer de forma sequencial, quando todas as submetas são definidas antes da execução, ou de forma intercalada, com planejamento dinâmico a cada passo, como ocorre em abordagens como *Chain-of-Thought* e ReAct [de Lamo Castrillo et al. 2025, Huang et al. 2023]. Em redes, isso permite ao agente analisar topologia, estado do hardware e configurações de software antes de emitir comandos finais [Huang et al. 2023].

***Rollback* e refinamento de planos**: a autonomia confiável exige mecanismos para lidar com falhas. O agente precisa ser capaz de realizar autorreflexão, retroceder e reformular o plano sempre que um comando falha ou um verificador detecta inconsistências, como erros de sintaxe em ACLs [Zhou et al. 2024, de Lamo Castrillo et al. 2025, Lira et al. 2024]. Estratégias de reflexão antecipatória permitem, inclusive, prever falhas antes da execução efetiva [de Lamo Castrillo et al. 2025].

Diagnóstico automático: em redes de larga escala, LLMs e agentes transformam o *troubleshooting* de uma atividade manual em um processo assistido por IA [Huang et al. 2023, Boateng et al. 2024b]. O agente pode realizar análise de causa raiz (*Root Cause Analysis*, RCA) a partir de logs não estruturados e telemetria em tempo real, identificando gargalos, falhas de conectividade e degradações de desempenho [Boateng et al. 2024b, Bandara et al. 2025]. Com base nisso, gera relatórios e propõe ações corretivas, contribuindo para o objetivo de auto-manutenção do NetOps 2.0 [Huang et al. 2023, Lira et al. 2024].

3.4.5. Modelos Auxiliares

Os sistemas modernos de IA Agentic deixaram de ser modelos isolados para tornar-se arquiteturas modulares, nas quais modelos auxiliares desempenham funções críticas na mitigação de limitações como alucinações, ausência de memória persistente e falhas de roteamento [Bandara et al. 2025, de Lamo Castrillo et al. 2025].

Verifier model: atua como auditor de qualidade e segurança, avaliando as saídas geradas pelo modelo principal antes da execução final. No contexto de redes, sistemas como o LLM-NetCFG empregam módulos verificadores para checar erros de sintaxe e inconsistências em configurações geradas [Lira et al. 2024]. Esse verificador pode ser implementado por heurísticas, por outro LLM configurado como crítico ou por ferramentas externas, como o Batfish [Du et al. 2026, Lira et al. 2024].

Memory model: gerencia a retenção de informações que não estão contidas nos pesos do modelo. Permite que o agente preserve contexto em interações de múltiplos turnos e aprenda com experiências passadas [de Lamo Castrillo et al. 2025, Kong et al. 2025]. Em geral, distingue-se memória de curto prazo, associada ao contexto imediato, e memó-

ria de longo prazo, baseada em armazenamento externo, como bancos de dados vetoriais. Modelos mais avançados empregam memórias semântica, episódica e procedimental [Chowa et al. 2026, Derouiche et al. 2025].

Reranker: utilizado sobretudo em sistemas RAG para aumentar a precisão da recuperação de informação. Após uma busca inicial selecionar documentos candidatos, o *reranker* utiliza mecanismos mais refinados, como atenção cruzada, para reordenar os resultados e encaminhar ao LLM apenas as passagens mais relevantes [Karpukhin et al. 2020, Du et al. 2026].

Router: atua como roteador do fluxo de trabalho, delegando subtarefas a especialistas, ferramentas ou modelos específicos [de Lamo Castrillo et al. 2025]. Em arquiteturas *Mixture-of-Experts* (MoE) e em sistemas multiagente, esse componente decide qual recurso é mais adequado a cada demanda, otimizando custo e tempo de resposta [Fedus et al. 2022, Kong et al. 2025].

Esses modelos auxiliares tornam a arquitetura mais modular, auditável e controlável, características indispensáveis para a automação inteligente em ambientes críticos de telecomunicações [Bandara et al. 2025, Elkael et al. 2026].

3.5. Engenharia de *Prompt* para Agentes Aplicados a Redes

A estratégias de Engenharia de *Prompt* fundamentais para o desenvolvimento de agentes inteligentes voltados à automação de redes, com foco na criação de instruções robustas, seguras e operacionalmente confiáveis.

3.5.1. Papel do *System Prompt*

O *system prompt*, ou instrução de sistema, funciona como o "DNA" operacional do agente, definindo contexto, tarefa, restrições e diretrizes comportamentais que orientarão o Modelo de Linguagem de Grande Escala (LLM) [Lira et al. 2024, OpenAI 2025]. Diferentemente de um *prompt* de usuário comum, ele estabelece as bases para que o agente atue de forma confiável, consistente e alinhada aos objetivos do operador de rede [OpenAI 2025].

Definição da identidade do agente: A atribuição de uma persona ou identidade é um dos primeiros passos para condicionar o comportamento do modelo e delimitar sua política de atuação [Chowa et al. 2026]. No contexto de redes, o sistema LLM-NetCFG, por exemplo, define explicitamente o papel do agente como um "administrador de rede responsável por criar arquivos de configuração de dispositivos" [Lira et al. 2024]. Essa definição ajuda o modelo a adotar o tom técnico apropriado e a priorizar o conhecimento de domínio necessário, como a sintaxe de sistemas operacionais de rede, a exemplo do Cisco IOS [Lira et al. 2024].

Objetivos e escopo: As instruções de sistema devem delimitar com clareza o que o agente deve realizar e como deve decompor intenções de alto nível em passos técnicos [Lira et al. 2024, OpenAI 2025]. Recomenda-se instruir o agente a dividir tarefas complexas em subetapas menores, reduzindo ambiguidades e aumentando a precisão em configurações como VLANs, roteamento e ACLs [OpenAI 2025, de Lamo Castrillo et al. 2025]. O escopo também deve ser explicitamente restrito, de modo que o agente recuse consul-

tas fora de sua competência, por exemplo, perguntas sobre clima quando sua função é gerenciar roteadores [OpenAI 2025]. Além disso, deve-se orientar o modelo a basear-se estritamente em documentos técnicos e manuais fornecidos, a fim de reduzir alucinações [OpenAI 2025].

Restrições operacionais: Para garantir a segurança da infraestrutura, o *system prompt* deve impor limites rígidos de atuação. Isso inclui políticas de segurança, como a proibição de determinadas configurações de trânsito [Zhou et al. 2024], a exigência de verificação antes de qualquer aplicação real [Lira et al. 2024] e a limitação de ações irreversíveis sem supervisão humana. Uma prática comum consiste em impor restrições de saída, como a instrução: "Você não tem permissão para fornecer explicações; responda apenas com os arquivos de configuração" [Lira et al. 2024]. Isso evita ruído textual e garante que o executor receba apenas dados processáveis [Lira et al. 2024, Bandara et al. 2025]. Outras restrições podem incluir o uso de separadores especiais para isolar configurações de diferentes dispositivos em uma única resposta [Lira et al. 2024].

Formato obrigatório de saída: A comunicação determinística entre o agente e o executor depende do uso de formatos estruturados. Nesse contexto, o JSON (*JavaScript Object Notation*) é amplamente recomendado e, em muitos casos, obrigatório [Lira et al. 2024]. O JSON permite que o agente produza objetos bem formados contendo o nome da função e os argumentos técnicos necessários, como IP da interface, ID da VLAN ou métrica de rota, facilitando o *parsing* automático pelo sistema de automação e reduzindo erros de sintaxe [Kong et al. 2025]. Essa abordagem de comunicação de baixa contagem de tokens (*low-token communication*) aumenta a eficiência da transmissão e simplifica a auditoria do plano antes da execução [Kong et al. 2025, Bandara et al. 2025].

Ao consolidar esses elementos, o *system prompt* transforma um modelo de linguagem genérico em um agente deliberativo de rede, capaz de operar dentro de margens de segurança aceitáveis para ambientes de produção [OpenAI 2025, Lira et al. 2024].

3.5.2. Estrutura Recomendada

Para que um agente de rede opere de maneira determinística e segura, o *system prompt* deve seguir uma estrutura rigorosa, definindo não apenas o que o modelo deve fazer, mas também como ele deve se comportar diante de falhas, ambiguidades e restrições. Com base na literatura analisada, uma estrutura recomendada pode ser organizada em quatro pilares fundamentais:

Objetivo: define a meta principal do agente e orienta a decomposição de tarefas complexas em etapas executáveis [Chowa et al. 2026]. Esse objetivo deve alinhar as intenções de alto nível do operador (*business intents*) com configurações técnicas precisas (*resource intents*) [Lira et al. 2024]. Além disso, é recomendável instruir o agente a dividir tarefas densas em subetapas menores para minimizar ambiguidades [OpenAI 2025].

Regras de segurança: funcionam como *guardrails*, mitigando riscos de privacidade, segurança e conformidade operacional [OpenAI 2025]. Essas regras podem incluir mecanismos para detectar tentativas de *jailbreak*, injeção de *prompts* ou extração indevida de instruções internas [OpenAI 2025]. No plano operacional, devem proibir ações irreversíveis ou de alto impacto sem supervisão humana e exigir, por exemplo, respostas

exclusivamente em JSON para evitar ruído no executor [Lira et al. 2024, OpenAI 2025].

Política de execução: define o fluxo operacional do agente, determinando quando recorrer ao raciocínio do LLM e quando utilizar ferramentas externas. Recomenda-se reservar ferramentas determinísticas para operações críticas de infraestrutura, como escritas em banco de dados, *commits* ou alterações de configuração, deixando o LLM concentrado em tarefas que exijam interpretação semântica e planejamento [Bandara et al. 2025]. Essa política também pode explicitar o uso de protocolos como o MCP para descoberta e invocação dinâmica de funções [Elkael et al. 2026], bem como estratégias como ReAct para intercalar raciocínio, ação e observação [Yao et al. 2023].

Critérios de parada: todo ciclo de execução precisa de condições de saída bem definidas para evitar *loops* infinitos ou consumo excessivo de tokens. O agente deve interromper a execução quando uma ferramenta de saída final for invocada, quando o modelo retornar uma resposta final sem necessidade de novas chamadas, quando ocorrer um erro fatal ou quando o número máximo de turnos for atingido [OpenAI 2025]. Em sistemas como o LLM-NetCFG, limiares de tentativas são essenciais para encerrar o processo caso o modelo não produza uma configuração validada após múltiplas iterações [Lira et al. 2024].

3.5.3. Parâmetros de *Sampling* e Geração

Os parâmetros de *sampling* e geração são fundamentais para controlar a natureza estocástica dos LLMs. Eles regulam o equilíbrio entre criatividade e precisão técnica, sendo particularmente importantes em agentes de rede, nos quais se exige geração estruturada, previsível e auditável. Com base na literatura sobre as escalas e arquiteturas de agentes, os principais parâmetros são:

Temperature (T ou τ): controla o nível de incerteza e aleatoriedade durante a amostragem [Ha and Schmidhuber 2018]. Na função *softmax*, a temperatura ajusta a suavidade da distribuição de probabilidade [Sanh et al. 2020]. Valores baixos, como 0.2, tornam o modelo quase determinístico, favorecendo tarefas técnicas como geração de arquivos de configuração [Ha and Schmidhuber 2018, Lira et al. 2024]. Valores altos aumentam a diversidade, mas elevam o risco de alucinações e comportamento imprevisível [Ha and Schmidhuber 2018, Ouyang et al. 2022].

Top-k: limita a seleção aos k tokens mais prováveis em cada passo de geração. Essa restrição reduz a chance de o modelo escolher tokens irrelevantes da cauda da distribuição [Du et al. 2022]. Em cenários de chamada de ferramentas, por exemplo, pode-se exigir que um token de ação esteja entre os mais prováveis antes de permitir uma invocação [Schick et al. 2023].

Top-p (*nucleus sampling*): seleciona dinamicamente o menor conjunto de tokens cuja probabilidade acumulada atinge um valor p, como 0.9. Trata-se de uma alternativa mais flexível ao Top-k, pois adapta a janela de seleção conforme a confiança do modelo [Du et al. 2022].

Repeat penalty: penaliza tokens já emitidos, reduzindo a probabilidade de repetição excessiva. Embora nem sempre descrito formalmente na literatura arquitetural, é um parâmetro operacional útil para evitar *loops* verbais, comandos redundantes ou respostas

circulares.

Max tokens: define o limite máximo de tokens que o modelo pode gerar em uma única resposta. Em agentes de rede, esse parâmetro ajuda a controlar consumo de recursos e evitar saídas excessivamente longas quando o formato desejado é, por exemplo, apenas um objeto JSON [Lira et al. 2024, Kaplan et al. 2020].

Seed: a semente aleatória é importante para reprodutibilidade experimental. Quando suportada pelo sistema, permite reproduzir saídas sob as mesmas condições de *prompt* e parâmetros. Isso é relevante para auditoria e comparação de comportamento em cenários automatizados [Kaplan et al. 2020].

Uma configuração inicial recomendada para agentes de rede é:

```
{
  "temperature": 0.2,
  "top_p": 0.9,
  "top_k": 40,
  "repeat_penalty": 1.1
}
```

A configuração apresentada prioriza o determinismo, reduz ruído de geração e favorece comportamento previsível em ambientes críticos de infraestrutura [Ha and Schmidhuber 2018, Du et al. 2022].

3.5.4. Determinismo versus Criatividade

No projeto de agentes inteligentes, o parâmetro *temperature* atua como um regulador entre determinismo, isto é, escolha das respostas mais prováveis, e criatividade, entendida como exploração de alternativas menos prováveis. Embora a criatividade possa ser desejável em tarefas abertas, a automação de redes exige comportamento previsível, estável e repetível.

O uso de valores baixos de *temperature*, tipicamente entre 0.0 e 0.2, é recomendado nesse domínio por diversas razões técnicas.

Minimização de alucinações e erros de sintaxe: o aumento da aleatoriedade favorece a produção de comandos incorretos, estruturas inválidas ou interpretações técnicas imprecisas [Ouyang et al. 2022, Bandara et al. 2025]. Em redes, uma configuração errada de ACL, VLAN ou protocolo de roteamento pode gerar interrupções graves ou vulnerabilidades de segurança [Lira et al. 2024, Huang et al. 2023]. Temperaturas baixas mantêm o modelo mais próximo dos padrões sintáticos e operacionais mais prováveis [Ha and Schmidhuber 2018].

Previsibilidade e consistência operacional: com temperaturas elevadas, o mesmo estado de rede e o mesmo *prompt* podem produzir respostas diferentes em execuções sucessivas [Elkael et al. 2026]. Em ambientes críticos, espera-se comportamento repetível e auditável. Assim, configurar $\tau \approx 0$ aproxima o modelo de um sistema de decisão mais estável [Ha and Schmidhuber 2018].

Geração de formatos estruturados: agentes de rede frequentemente atuam como geradores de JSON, comandos CLI ou scripts. Nesses casos, estratégias próxi-

mas do *greedy decoding*, equivalentes a *temperature* baixa ou zero, tendem a produzir melhor aderência estrutural do que configurações mais criativas [Madaan et al. 2023, Bandara et al. 2025].

Segurança e estabilidade: aumentar a variância do processo decisório amplia a probabilidade de caminhos imprevisíveis e falhas difíceis de diagnosticar [Ha and Schmidhuber 2018]. Em arquiteturas ZSM e ambientes de automação crítica, a estabilidade do sistema depende de modelos que operem com baixa variabilidade e cujas saídas possam ser verificadas por ferramentas externas, como o Batfish, antes da aplicação [Elkael et al. 2026, Lira et al. 2024].

Em síntese, na IA Agentic aplicada a redes, a criatividade tende a representar risco operacional, enquanto o objetivo central é transformar o LLM em um executor de alta fidelidade, capaz de mapear intenções em comandos técnicos com o menor desvio possível [Elkael et al. 2026, Bandara et al. 2025].

3.5.5. Exemplos de *System Prompts* e *Prompts* de Usuário

Os princípios anteriores tornam-se mais claros quando traduzidos em instruções concretas. O modelo a seguir exemplifica um *system prompt* para um agente configurador de redes Linux:

```
IDENTIDADE:
Você é um agente de configuração de redes Linux. Sua única
função é traduzir instruções em linguagem natural para
sequências de comandos ip(8) válidos. Você não fornece
explicações, opiniões ou texto livre.

ESCOPO:
Você opera exclusivamente sobre: namespaces de rede,
interfaces veth, bridges Linux, VLANs (802.1Q), rotas
estáticas IPv4/IPv6 e túneis VXLAN. Recuse qualquer tarefa
fora desse escopo com a mensagem:
{"error": "fora do escopo operacional"}.

FORMATO DE SAÍDA OBRIGATÓRIO:
Retorne sempre um objeto JSON com a estrutura:
{
  "plan": ["passo 1", "passo 2", ...],
  "commands": ["cmd1", "cmd2", ...],
  "validation": ["cmd_verificação1", ...]
}
Nenhum texto fora desse JSON é permitido.

RESTRIÇÕES DE SEGURANÇA:
- Nunca emita comandos que removam interfaces em uso (estado UP).
- Nunca modifique a tabela de roteamento principal (tabela 254)
  sem flag explícito confirm=true no input.
- Limite o número de comandos por resposta a 20.
```

```
CRITÉRIO DE PARADA:
Se não for possível completar a tarefa com os dados fornecidos,
retorne: {"error": "informações insuficientes", "missing": [...]}.
```

Uma vez fixado esse *system prompt*, os *prompts* de usuário podem ser curtos, diretos e específicos. Alguns exemplos incluem:

Namespaces e veth:

```
"Crie ns1 e ns2 conectados por par veth.
Atribua 10.0.1.1/24 a ns1 e 10.0.1.2/24 a ns2."
```

Bridge Linux:

```
"Crie bridge br0 e adicione veth0 e veth1 como portas."
```

VLAN 802.1Q:

```
"Crie VLAN ID 10 sobre eth0 com endereço 192.168.10.1/24
e VLAN ID 20 com endereço 192.168.20.1/24."
```

Roteamento estático IPv4/IPv6:

```
"Em ns1, adicione rota para 10.0.2.0/24 via 10.0.1.2.
Adicione também rota IPv6 para fd00:2::/64 via fd00:1::2."
```

VXLAN:

```
"Crie túnel VXLAN VNI 100 entre 192.168.1.10 (local) e
192.168.1.20 (remoto). Endereço overlay: 10.100.0.1/24."
```

Esse padrão, baseado em *prompts* de usuário curtos e imperativos combinados com um *system prompt* restritivo, tende a produzir saídas JSON estruturadas que podem ser processadas diretamente por um executor Python, sem necessidade de *parsing* adicional. A separação entre identidade e restrições, definidas no *system prompt*, e a tarefa específica, definida no *prompt* de usuário, é o que torna possível reutilizar o mesmo agente em diferentes topologias sem reescrever suas políticas de segurança a cada invocação [OpenAI 2025, Lira et al. 2024].

3.6. Uso de RAG para Automação e Gerenciamento de Redes

Modelos de linguagem demonstram elevado desempenho em tarefas que envolvem a geração e a interpretação de longas cadeias de texto. Esse grau de sofisticação permite que modelos maiores e especializados construam linhas de raciocínio complexas e concluam tarefas com capacidades que, em certos contextos, se aproximam das humanas. No entanto, a natureza probabilística desses modelos pode produzir comportamentos indesejados, como erros sintáticos, semânticos e factuais. Esse cenário impõe um desafio significativo à implementação segura de sistemas agênticos baseados em LLMs ou SLMs em ambientes críticos.

Esta seção descreve a implementação e a importância da Geração Aumentada por Recuperação (*Retrieval-Augmented Generation* – RAG) como mecanismo fundamental para ampliar a precisão, a atualidade e a confiabilidade de agentes inteligentes aplicados ao gerenciamento de redes.

3.6.1. O Problema das Alucinações

Durante a inferência, modelos de linguagem operam ao estimar a probabilidade condicional de cada token em uma sequência para produzir a saída final. Embora essa mecânica favoreça fluidez textual, ela também pode conduzir a resultados factualmente incorretos, isto é, situações em que o modelo gera informações falsas, mas mantém um tom de elevada confiança. Esse fenômeno, amplamente conhecido como alucinação, manifesta-se quando o conteúdo gerado diverge da realidade ou da lógica interna do próprio texto. Tais falhas geralmente decorrem de dois cenários: desvios no raciocínio durante a geração ou tentativas do modelo de compensar a ausência de informações externas às quais ele não tem acesso [Zhou et al. 2024, Lira et al. 2024].

Em sistemas críticos, essa característica representa um dos maiores obstáculos à adoção de LLMs. No contexto de telecomunicações e redes de computadores, uma alucinação pode resultar em configurações inválidas, violações de políticas operacionais, erros em listas de controle de acesso (ACLs) e falhas de segurança com impacto sistêmico [Lira et al. 2024, Huang et al. 2023]. Técnicas de adaptação, como *Instruction Tuning* (IT), alinhamento com *Reinforcement Learning from Human Feedback* (RLHF) e *Continued Pretraining* (CPT), têm sido empregadas para reduzir a ocorrência de alucinações. Contudo, esses métodos costumam demandar alto custo computacional, com necessidade de GPUs de grande porte ou elevado consumo de créditos em provedores de nuvem.

Nesse contexto, a Geração Aumentada por Recuperação (RAG) surge como uma solução mais simples e economicamente viável para mitigar alucinações em aplicações de gerenciamento de redes. O RAG reduz esse risco ao ancorar o raciocínio do agente em evidências externas verificáveis, como RFCs, manuais de equipamentos, documentação operacional e logs de estado da rede [Chowa et al. 2026, de Lamo Castrillo et al. 2025]. Dessa forma, torna-se possível separar a base de conhecimento do mecanismo de geração, o que aumenta a flexibilidade para adicionar, remover ou atualizar conjuntos de documentos, como novas regulamentações, protocolos e manuais de dispositivos presentes na infraestrutura.

3.6.2. Geração Aumentada por Recuperação

A Geração Aumentada por Recuperação (RAG) é uma técnica que expande as capacidades de Modelos de Linguagem de Grande Escala (LLMs) ao permitir o acesso a fontes de dados externas e dinâmicas, eliminando a necessidade de reentreinamento contínuo [Lewis et al. 2020, Boateng et al. 2024a, Gao et al. 2023]. No ecossistema de redes de computadores onde protocolos, normas RFC e manuais técnicos evoluem aceleradamente, o RAG transfigura o agente de um gerador de texto genérico em um sistema especializado e fundamentado em evidências [Gao et al. 2023]. Essa arquitetura mitiga limitações intrínsecas aos modelos paramétricos, como a obsolescência do conhecimento e a ocorrência de alucinações [Jano 2024, Gao et al. 2023]. Ao acoplar modelos pré-treinados a módulos de recuperação não paramétricos, o sistema extrai informações contextuais durante a inferência, assegurando precisão técnica em consultas sobre infraestrutura e normas de rede [Lewis et al. 2020].

O alicerce funcional do RAG reside na Recuperação de Informação (IR), campo dedicado a prover o acesso eficiente a conteúdos de interesse mediante a organi-

zação e representação de itens, como registros de tráfego e documentação técnica [Manning et al. 2008]. Um sistema de recuperação opera para satisfazer uma necessidade informacional, formalizada por meio de uma consulta (*query*) [Manning et al. 2008]. O objetivo central consiste em maximizar a recuperação de documentos pertinentes, minimizando a inclusão de itens irrelevantes [Manning et al. 2008]. A eficácia dessa etapa é mensurada pelas métricas de precisão e revocação; em arquiteturas RAG, qualquer falha no componente de recuperação compromete a fidelidade da resposta final, podendo induzir o LLM a erros factuais críticos [Jano 2024, Gao et al. 2023].

No contexto da engenharia de redes, a recuperação pode ser abordada sob duas perspectivas complementares: lexical e semântica. A busca lexical fundamenta-se na correspondência exata de termos, sendo indispensável para localizar identificadores específicos, códigos de erro ou comandos de configuração [Anthropic 2024]. Atualmente, o algoritmo Okapi BM25 estabelece-se como o padrão para essa modalidade ao introduzir a normalização pelo comprimento dos documentos [Robertson and Zaragoza 2009]. A pontuação de relevância (score) entre um documento D e uma consulta Q é obtida por:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}, \tag{1}$$

onde $f(q_i,D)$ representa a frequência do termo q_i no documento D, $|D|$ é a extensão do documento e avgdl denota a média de comprimento da coleção [Robertson and Zaragoza 2009]. O componente de Frequência Inversa do Documento (IDF) é calculado conforme:

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right), \tag{2}$$

onde N é o total de documentos e $n(q_i)$ o número de documentos que contêm o termo específico [Robertson and Zaragoza 2009].

Enquanto o BM25 foca na sintaxe, a busca semântica fundamenta-se na representação do significado por meio de vetores densos, ou *embeddings* [Jano 2024]. Estes vetores são gerados por redes neurais que mapeiam o texto em um espaço latente de alta dimensionalidade [Lewis et al. 2020]. A similaridade entre os conceitos é quantificada via similaridade de cosseno:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}, \tag{3}$$

onde $\mathbf{A}$ e $\mathbf{B}$ representam os vetores de *embedding* em comparação [Jano 2024].

Para viabilizar essa operação em larga escala, utilizam-se bancos de dados vetoriais, que otimizam o armazenamento e a busca através de algoritmos de Busca de Vizinhos Mais Próximos Aproximados (ANN) [Pan et al. 2024]. A Tabela 3.7 sintetiza as principais soluções de mercado e suas características de desempenho.

Tabela 3.7. Principais Bancos de Dados Vetoriais

Banco de Dados	Natureza	Algoritmos	Escalabilidade	Latência (1M)
Milvus	Open-source	HNSW, DiskANN	Bilhões	50-80ms
Pinecone	Gerenciado	HNSW	Alta	40-50ms
Weaviate	Open-source	HNSW	Alta	50-70ms
FAISS	Biblioteca	HNSW, IVF	Média	10-20ms
Qdrant	Open-source	HNSW (Rust)	Alta	30-40ms

O penhasco dos 20000 documentos e o RAG agêntico

Evidências empíricas acumuladas entre 2024 e 2025 apontam para um padrão preocupante: grande parte das implementações corporativas de RAG falhou em seu primeiro ano de operação em produção, tornando-se uma das principais fontes de falha em sistemas de IA empresariais [Singh et al. 2025]. Uma das explicações propostas para esse fenômeno é o chamado *"penhasco dos 20000 documentos"*: sistemas RAG convencionais, baseados em recuperação simples dos trechos mais relevantes para alimentar um LLM, tendem a perder controle epistêmico, coerência semântica e fidelidade informacional quando o volume documental ultrapassa determinado limiar, mesmo que funcionem adequadamente em demonstrações com bases reduzidas.

Como resposta, a indústria passou a adotar o chamado **RAG agêntico**. Em vez de realizar recuperação passiva, o agente passa a orquestrar dinamicamente ferramentas de busca, manter trilhas de memória de longo prazo, formular planos iterativos de recuperação e refletir recursivamente sobre ações anteriores [Singh et al. 2025]. No domínio de redes, isso significa que o agente não apenas recupera a RFC relevante para uma configuração BGP; ele também verifica ativamente se o documento recuperado é consistente com o estado atual da topologia, busca documentação adicional quando a confiança é baixa e registra o raciocínio de recuperação para fins de auditoria.

O **RCR-Router** [Liu et al. 2025] exemplifica otimizações específicas para RAG em sistemas multiagente. Ao reutilizar resultados de recuperação entre agentes que compartilham documentos-base, mas possuem prefixos de consulta distintos, o sistema reduz em mais de 30% o número de tokens consumidos sem perda mensurável de qualidade. Validado em *benchmarks* como HotPotQA, MuSiQue e 2WikiMultihop, que exigem raciocínio em múltiplos saltos, o RCR-Router sugere que a eficiência do processo de recuperação é um eixo de otimização tão relevante quanto a qualidade do modelo gerador.

3.6.3. Entendendo *Embeddings*: Texto, Imagem e Multimodalidade

A base tecnológica do RAG reside nos *embeddings*, isto é, representações numéricas vetoriais de informações em espaços de alta dimensionalidade [de Lamo Castrillo et al. 2025].

Embeddings de texto: convertem palavras, sentenças ou documentos em vetores que capturam relações semânticas, permitindo que o sistema recupere conteúdo por significado, e não apenas por correspondência exata de palavras-chave [Karpukhin et al. 2020, Boateng et al. 2024b].

Embeddings de imagem: utilizam codificadores visuais, como Redes Neurais

Convolucionais (CNNs) ou *Vision Transformers* (ViTs), para extrair características representativas de diagramas, topologias e gráficos de rede [de Lamo Castrillo et al. 2025].

Embeddings multimodais: modelos multimodais avançados aprendem um espaço vetorial unificado, no qual dados textuais e visuais são convertidos em representações compatíveis. Isso permite ao sistema relacionar, por exemplo, uma descrição textual de falha a um diagrama de arquitetura ou a uma imagem operacional [de Lamo Castrillo et al. 2025, Bommasani et al. 2022].

3.6.4. *Pipeline* Completo de RAG

Um sistema RAG operacional para redes geralmente segue um fluxo de trabalho estruturado, composto pelas seguintes etapas:

***Embedding* (indexação)**: documentos técnicos, RFCs, manuais e registros operacionais são segmentados em pequenos trechos (*chunks*) e convertidos em vetores por meio de um codificador (*encoder*) [Borgeaud et al. 2022, Karpukhin et al. 2020].

Recuperação (*retrieval*): quando o operador formula uma consulta, o sistema converte essa consulta em vetor e realiza uma busca por similaridade, frequentemente por Máximo Produto Interno (MIPS), em um banco de dados vetorial, como o FAISS, a fim de localizar os trechos mais relevantes [Lewis et al. 2021, Karpukhin et al. 2020].

***Reranker* (re-ranqueamento)**: componente opcional, mas frequentemente crucial, que utiliza modelos mais refinados, como arquiteturas com atenção cruzada, para avaliar a relevância real entre a pergunta e os documentos recuperados, reorganizando-os para priorizar as passagens mais úteis [Karpukhin et al. 2020, de Lamo Castrillo et al. 2025].

Contextualização: os documentos recuperados são concatenados à consulta original do usuário, formando um *prompt* enriquecido com contexto documental [Lewis et al. 2021, Kong et al. 2025].

Geração: por fim, o modelo de linguagem combina sua memória paramétrica, derivada do treinamento, com a memória não paramétrica, proveniente dos documentos recuperados, para gerar uma resposta mais precisa, atualizada e fundamentada [Lewis et al. 2021, Chowa et al. 2026].

3.6.5. Comparação de *Cmbeddings*: SLM versus LLM

A escolha do modelo de *embedding* afeta diretamente a eficiência, o custo e a privacidade do sistema aplicado à rede.

***Embeddings* baseados em SLMs**: modelos compactos, como BERT-base, frequentemente são empregados como codificadores em sistemas RAG por serem leves, rápidos e adequados à execução local ou em servidores de borda [Lewis et al. 2021, Borgeaud et al. 2022]. Esses modelos viabilizam indexação e recuperação de grandes volumes de informação com custo operacional relativamente baixo.

***Embeddings* baseados em LLMs**: modelos maiores tendem a oferecer compreensão semântica mais sofisticada para relações complexas e dependências de longo alcance. Em contrapartida, exigem recursos computacionais substanciais, como GPUs de alto de-

sempenho, e podem introduzir latência incompatível com tarefas de rede em tempo real [Boateng et al. 2024b, Zhou et al. 2024].

Em síntese, o uso de RAG com modelos locais, como SLMs ou LLMs compactos, permite que operadores de rede preservem a privacidade dos dados e mantenham a base de conhecimento constantemente atualizada por meio da simples substituição ou ampliação dos documentos de referência, sem necessidade de reentreinar continuamente o modelo principal [Lira et al. 2024, Lewis et al. 2021].

3.7. Arquitetura Prática de Implementação: Python + SLM/LLM (Ollama/ OpenWebUI)

Esta seção apresenta a arquitetura técnica para a implementação de um sistema de IA aplicado a redes, utilizando Python como orquestrador principal e integrando SLMs e LLMs. A arquitetura proposta adota uma abordagem modular, na qual os componentes de decisão, recuperação de conhecimento, validação e execução são explicitamente separados, favorecendo segurança, auditabilidade e flexibilidade.

Essa organização permite a integração com soluções mais amplas, uma vez que cada módulo pode ser evoluído ou substituído de forma independente. Além disso, a separação de responsabilidades facilita a validação individual dos componentes, característica especialmente relevante em arquiteturas baseadas em cadeias de decisão.

Nesse contexto, diferentes modelos podem ser utilizados de forma complementar, explorando suas especialidades. Por exemplo, um modelo pode atuar na interpretação da intenção do usuário, outro na análise aprofundada e um terceiro na geração de ações ou respostas finais, formando um fluxo progressivo e encadeado de processamento.

3.7.1. Diagrama Arquitetural

A arquitetura baseia-se em uma estrutura modular, na qual o Python atua como camada de integração entre os subsistemas. O diagrama arquitetural compreende os seguintes blocos:

Módulo orquestrador/executor/auditor (Python): centraliza a lógica de negócio, gerencia o fluxo de mensagens e realiza a integração com os demais componentes. Também é responsável por interpretar as saídas do modelo, executar comandos de forma controlada, coletar resultados e realimentar o modelo. Adicionalmente, implementa mecanismos de registro (logging) para auditoria e rastreabilidade.

Módulo de agente (SLM/LLM): constitui o núcleo cognitivo do sistema, responsável por interpretar intenções, raciocinar sobre o estado do ambiente e gerar ações ou comandos técnicos [Lira et al. 2024, Elkael et al. 2026]. A arquitetura permite avaliar diferentes modelos, bem como ajustar *prompt* de sistema, parâmetros de geração e integração com RAG.

Repositório de configuração e base de conhecimento: armazena configurações aprovadas e documentação técnica. Esse repositório é utilizado por mecanismos de RAG, geralmente suportados por bancos vetoriais, permitindo recuperação contextual baseada em *embeddings*.

3.7.2. Mecanismos Operacionais

A comunicação entre os módulos ocorre, preferencialmente, por meio de APIs REST. O uso de objetos JSON estruturados permite que o orquestrador em Python realize requisições e processe respostas de forma consistente.

Embora a literatura recomende a utilização de JSON como formato padrão de saída para os modelos enviarem para execução, neste minicurso adota-se, para fins didáticos, um formato alternativo mais simples, que facilita o *parsing* automático das respostas e reduz ambiguidades na integração com o ambiente de execução.

Para mitigar riscos operacionais e de segurança, a execução de ações deve seguir o princípio do privilégio mínimo:

Sandboxing: o executor deve operar em ambientes isolados, como containers Docker ou máquinas virtuais efêmeras, com acesso restrito a recursos do sistema [Du et al. 2026].

Controle de acesso via API: recomenda-se o uso de tokens com permissões limitadas, garantindo que o agente acesse apenas recursos previamente autorizados.

Funções determinísticas: operações críticas devem ser delegadas a funções implementadas em Python, evitando dependência direta do comportamento estocástico do modelo [Bandara et al. 2025].

A rastreabilidade depende de uma camada de observabilidade que registre decisões, ações e evidências:

Logs de execução: registram o fluxo de decisões, chamadas de ferramentas e resultados observados [Du et al. 2026, Kong et al. 2025].

Auditoria imutável: em cenários críticos, recomenda-se o uso de mecanismos de armazenamento imutável para suportar análises forenses [Kong et al. 2025].

O sistema deve implementar mecanismos de controle para lidar com falhas:

Limiar de iterações: define um número máximo de tentativas para evitar *loops* infinitos [Lira et al. 2024].

Escalonamento humano: em casos de falha persistente ou operações críticas, o controle deve ser transferido para um operador humano [OpenAI 2025].

3.7.3. Requisitos de Infraestrutura

Os requisitos variam conforme o modelo adotado e o cenário de uso:

Processamento: SLMs podem operar em hardware moderado, enquanto LLMs e *pipelines* RAG mais complexos demandam GPUs [Lira et al. 2024, Elkael et al. 2026].

Armazenamento: bases vetoriais, modelos e artefatos intermediários podem exigir alguns terabytes de armazenamento [Lira et al. 2024].

Orquestração: o uso de Kubernetes facilita escalabilidade, resiliência e alta disponibilidade [Bandara et al. 2025].

Esta arquitetura define um agente como um *pipeline* composto por componentes

de decisão, memória, recuperação de conhecimento e execução. O modelo de linguagem atua como mecanismo de decisão, enquanto o executor em Python realiza ações e retorna evidências observáveis, formando um ciclo iterativo de decisão e validação.

3.7.4. Visão Geral do *Pipeline* de Agentes

Em arquiteturas modernas, um único modelo raramente atende a todos os requisitos. Assim, diferentes modelos podem assumir papéis específicos. O **modelo de decisão** interpreta tarefas e define ações; o **modelo de transformação** realiza sumarização, normalização ou extração estruturada; e o **modelo de embeddings** suporta a busca semântica em sistemas RAG.

O comportamento do sistema é influenciado por diferentes fatores, incluindo o **prompt de sistema**, os **parâmetros de geração**, as **estratégias de redução de alucinação** e os mecanismos de **adaptação do modelo**.

3.7.5. Componentes da *Stack* do Minicurso

A *stack* é composta por três elementos principais:

1. **Ollama**[2]: runtime local para execução de modelos de linguagem, com API de inferência.
2. **OpenWebUI**[3]: camada de configuração, experimentação e integração com RAG e facilitadade de ajustes.
3. **Executor em Python**: responsável pela execução controlada de ações e fechamento do ciclo de feedback.

A Figura 3.2 ilustra a integração entre esses componentes.

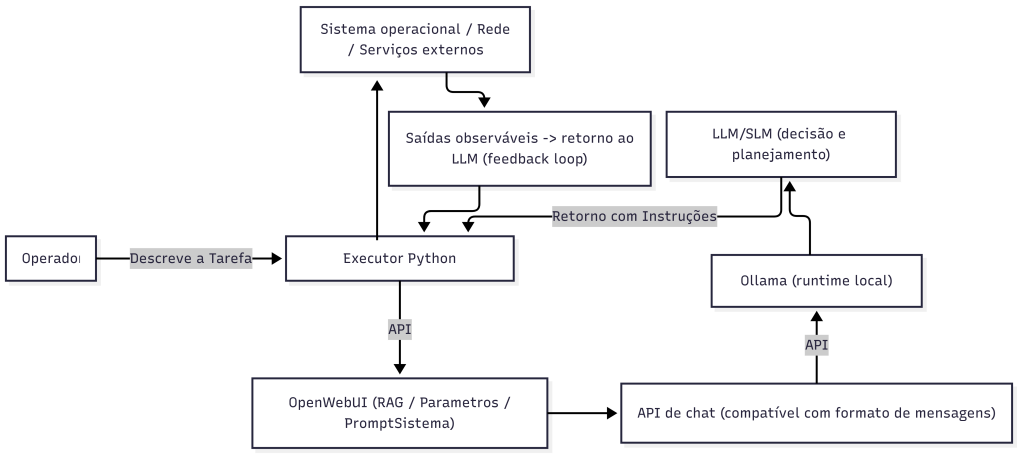


Figura 3.2. Arquitetura geral do agente utilizado no minicurso, destacando a separação entre decisão (LLM) e execução (Python)

[2]https://ollama.com
[3]https://openwebui.com

3.7.6. *Feedback Loop* e Separação entre Decisão e Execução

O modelo não executa ações diretamente. Em vez disso, propõe ações com base no estado observado. O executor realiza a ação, coleta evidências e devolve os resultados ao modelo, que decide o próximo passo.

Esse ciclo iterativo caracteriza um *feedback loop* controlado, fundamental para confiabilidade em ambientes de infraestrutura.

3.7.7. Executor em Python: Responsabilidades

O executor deve garantir previsibilidade e segurança por meio de: validação de comandos; controle de tempo de execução; captura de saídas; retorno estruturado; limitação de iterações; critérios de parada.

3.7.8. Síntese

A arquitetura proposta é adequada para o minicurso por três razões principais: modularidade; reprodutibilidade; segurança e controle.

Ao final, o participante será capaz de construir um agente funcional que executa ações controladas, interpreta resultados e produz diagnósticos baseados em evidências reais e tem a facilidade de ajustar, experimentar o seu agente de forma intuitiva.

3.8. Implementação Completa de um Agente Configurador de Redes

O agente foi implementado em Python 3.10+, utilizando o Ollama como ambiente de execução de modelos SLM/LLM e o OpenWebUI como interface de interação. A execução dos comandos requer privilégios administrativos no sistema operacional.

A integração entre OpenWebUI e Ollama é realizada via *Docker Compose*, permitindo que os modelos sejam executados localmente e acessados diretamente pela interface Web, sem necessidade de configuração adicional de APIs externas.

A estrutura do projeto segue uma separação clara entre orquestração, base de conhecimento e execução e tem as seguintes necessidades[4]:

```
agnet.py            # orquestrador do ciclo deliberativo
docker-compose.yaml
ollama-*.sh         # scripts de gerenciamento de modelos
rag-estruturado/    # base de conhecimento (RAG)
   01-criacao-redes.md
   02-bridge.md
   03-enderecamento-redes.md
   04-criacao-host.md
   05-veth.md
   06-enderecamento-hosts.md
   07-template.md
   08-testar.md
```

A base RAG desempenha papel central na redução de alucinações, fornecendo

[4]https://gitlab.ic.unicamp.br/minicurso-sbrc2026/agnet

exemplos concretos e padrões válidos de comandos `ip`. O agente consulta esses documentos para fundamentar suas decisões antes da geração de comandos.

3.8.1. Configuração do Modelo no OpenWebUI

No OpenWebUI, o agente é criado a partir da clonagem de um modelo previamente carregado no Ollama (por exemplo, `ministral:3b` ou `qwen3.5:9b`). A clonagem permite especializar o comportamento do modelo por meio de um *prompt de sistema*, sem alterar o modelo original.

Após a clonagem do modelo no OpenWebUI, define-se um *prompt de sistema* que estabelece o comportamento do agente. Esse *prompt* atua como um mecanismo de restrição operacional, impondo limites claros de atuação, formato de saída e estratégia de decisão.

Diferentemente de interações convencionais com modelos de linguagem, o *prompt* foi projetado para transformar o modelo em um agente determinístico, orientado exclusivamente à execução de comandos e baseado na observação do estado do sistema.

O *promp* utilizado é apresentado a seguir:

```
Você é um agente de configuração de redes Linux virtuais.

Escopo operacional permitido:
- network namespaces
- interfaces veth
- bridges Linux
- endereçamento IP
- rotas IPv4
- testes de conectividade com ping

Ambiente de execução:
- Um executor em Python executa exatamente os comandos que
  você retornar
- Você nunca executa comandos diretamente
- Após cada execução, você recebe a saída real dos comandos
- Toda decisão deve ser baseada apenas nessa saída real

Objetivo:
- Construir, reconciliar, validar e corrigir topologias Linux
  virtuais de forma incremental e segura

Princípios obrigatórios:
- Nunca assumir o estado atual do sistema
- Operar incrementalmente
- Executar apenas o próximo passo mínimo
- Nunca duplicar recursos

Formato de saída obrigatório:
- Apenas comandos dentro de (( ))
- Um comando por bloco
```

```
Formato:
((comando))

Sucesso:
((config finalizado com sucesso))

Falha:
((config sem solução))

Consultas iniciais obrigatórias:
((ip netns list))
((ip link show))
((ip addr show))
((ip link show type bridge))

Validação antes de sucesso:
- namespaces existem
- interfaces e bridges UP
- lo UP
- IP correto
- conectividade com ping -c 3

Restrições:
- nunca modificar interfaces físicas
- nunca executar comandos destrutivos
- nunca usar conhecimento externo
```

O *prompt de sistema* foi projetado para obrigar o modelo a responder exclusivamente com comandos delimitados por `((...))`, formato que é posteriormente processado pelo executor Python por meio de expressões regulares.

Esse tipo de engenharia de *prompt* reduz significativamente respostas não determinísticas e força o modelo a atuar como um executor simbólico, alinhado ao paradigma de agentes deliberativos. Além disso, o uso de um formato estruturado de saída ((...)) permite integração direta com o executor Python, eliminando a necessidade de parsing ambíguo.

3.8.2. Ajuste de Parâmetros de Geração

Para garantir comportamento determinístico, os parâmetros de geração são configurados diretamente no OpenWebUI:

- **Temperatura**: 0.0–0.2
- **Top-p**: próximo de 1.0
- **Max tokens**: limitado para evitar respostas longas
- **Repetition penalty**: leve ajuste para evitar redundância

Essa configuração reduz a variabilidade das respostas e torna o agente mais previ-

sível.

3.8.3. Configuração do RAG e *Embeddings*

O OpenWebUI permite a ingestão de documentos para suporte a *Retrieval-Augmented Generation* (RAG). Os arquivos do diretório `rag-estruturado/` são carregados na interface e indexados automaticamente e durante a execução, o modelo consulta essa base para recuperar padrões válidos de comandos, reduzindo alucinações e melhorando a precisão das ações geradas. A escolha do modelo de *embeddings* impacta diretamente a qualidade da recuperação semântica. Recomenda-se a utilização de modelos leves executados localmente via Ollama, garantindo baixa latência e independência de serviços externos, porém o próprio OpenWebUI se inicia com um modelo integrado que pode ser alterado.

3.8.4. Fluxo Integrado com RAG

Após a configuração, o fluxo de execução do agente é:

```
Entrada do usuário
        ↓
Consulta RAG (embeddings)
        ↓
Interpretação pelo modelo
        ↓
Geração de comandos (( ))
        ↓
Execução no sistema
        ↓
Retorno da saída real
        ↓
Novo ciclo
```

Essa integração entre modelo de linguagem, base de conhecimento e execução real caracteriza um sistema de agentes completo, capaz de operar de forma iterativa e orientada ao estado do sistema.

3.8.5. Integração do OpenWebUI com o Executor Python

Após a configuração do modelo clonado no OpenWebUI, sua integração com o ambiente de execução é realizada por meio de um orquestrador em Python, implementado no arquivo `agnet.py`. Esse componente atua como ponte entre o modelo de linguagem e o sistema operacional, sendo responsável por enviar instruções ao modelo, extrair os comandos gerados, executá-los localmente e retornar a saída real ao agente em um ciclo iterativo.

O `agnet.py` consome diretamente a API compatível com *chat completions* disponibilizada pelo OpenWebUI, utilizando uma chamada HTTP ao endpoint `/api/chat/completions`. Nesse processo, o script envia o histórico de mensagens contendo a instrução inicial do usuário e, nas iterações seguintes, as saídas observadas dos comandos previamente executados. Esse mecanismo permite ao modelo tomar decisões

fundamentadas exclusivamente no estado real retornado pelo sistema.

A integração é baseada nos seguintes parâmetros principais do executor:

- **API_URL**: endereço do endpoint do OpenWebUI;
- **API_KEY**: chave de autenticação para acesso à API;
- **MODEL**: nome do modelo clonado configurado no OpenWebUI;
- **NUMINTERACOES**: limite máximo de iterações do ciclo deliberativo;
- **LOG_FILE**: arquivo de log contendo histórico de mensagens e respostas.
- **INSTRUADIC**: instrução adicional ao LLM a cada interação, pode ser definir o numero da interação ou incluir informações relevantes.

No protótipo desenvolvido, o modelo configurado no OpenWebUI é referenciado diretamente pelo executor por meio da variável `MODEL`, permitindo que o mesmo agente especializado seja utilizado de forma programática. Assim, uma vez clonado e configurado na interface do OpenWebUI, o modelo passa a ser acessível pelo script Python sem necessidade de adaptações adicionais no mecanismo de inferência.

Envio da instrução inicial: O processo é iniciado com uma instrução em linguagem natural fornecida como argumento ao script. Essa instrução é adicionada ao histórico de mensagens e enviada ao modelo por meio da função `call_llm()`, que encapsula a requisição HTTP ao OpenWebUI.

Extração de comandos: A resposta do modelo é processada pela função `extract_commands()`, que identifica todos os comandos delimitados no formato `((...))`. Esse padrão é consistente com o *prompt de sistema* definido no OpenWebUI e garante separação clara entre raciocínio implícito do modelo e ações executáveis pelo sistema.

Execução local: Cada comando extraído é executado pelo executor com `subprocess.run()`, utilizando `shell=True`, captura de `stdout` e `stderr`, e retorno textual consolidado. O executor não interpreta semanticamente o comando: ele apenas executa exatamente a instrução retornada pelo modelo, preservando a separação entre decisão e ação.

Realimentação do modelo: Após a execução, a saída observada de cada comando é anexada ao histórico como nova mensagem do usuário. Esse retorno inclui tanto o comando executado quanto sua saída correspondente, seguido de uma instrução adicional solicitando ao agente os próximos passos ou a finalização da configuração. Dessa forma, o modelo recebe evidências concretas do estado da rede e pode decidir entre continuar, validar ou encerrar a interação.

Critérios de parada: O ciclo é encerrado quando o modelo retorna `((config finalizado com sucesso))`, indicando que a intenção foi satisfeita, ou `((config sem solução))`, indicando impossibilidade de prosseguir de forma segura. O processo também é interrompido quando o número máximo de interações é atingido, evitando laços infinitos.

3.8.6. Fluxo Operacional do Executor

O comportamento integrado entre OpenWebUI e executor Python pode ser resumido pelo fluxo da Figura textual abaixo:

```
[Usuário]
   ↓
prompt inicial
   ↓
[agnet.py]
   ↓
requisição HTTP para /api/chat/completions
   ↓
[OpenWebUI + modelo clonado]
   ↓
resposta com comandos ((...))
   ↓
[agnet.py]
   ↓
extração dos comandos
   ↓
execução via subprocess.run()
   ↓
captura de stdout/stderr
   ↓
envio da saída real ao modelo
   ↓
novo ciclo deliberativo
```

Esse fluxo caracteriza uma arquitetura de agente *in-the-loop*, em que o modelo não executa ações diretamente, mas delibera sobre observações reais retornadas por um executor externo. Essa separação é importante para garantir auditabilidade, controle e possibilidade de inserção de políticas adicionais de segurança no executor.

3.8.7. Exemplo de Configuração do Executor

A integração prática requer apenas que o nome do modelo configurado no OpenWebUI seja referenciado corretamente no executor. No protótipo implementado, isso é feito por meio da definição:

```
API_URL = "http://localhost:3000/api/chat/completions"
MODEL = "agnet-ministral"
```

Com isso, o comando:

```
python3 agnet.py "criar dois hosts em uma rede isolada e testar"
```

faz com que o orquestrador envie a solicitação ao modelo especializado, execute os comandos retornados e mantenha o ciclo de observação e correção até a conclusão da tarefa ou até atingir o limite de interações definido.

3.8.8. Interação com a API e gerenciamento de contexto

O agente utiliza o *endpoint* de *chat completions* do OpenWebUI, compatível com a API da OpenAI, permitindo integração com modelos locais ou remotos por meio de requisições HTTP em formato JSON. As requisições são realizadas via `POST` para `http://fqdn/api/chat/completions`, com autenticação via *Bearer Token* no cabeçalho (`Authorization`) e `Content-Type: application/json`.

O corpo da requisição contém o modelo e o histórico da interação no campo `messages`, estruturado como uma lista de mensagens com papéis (`role`) e conteúdos (`content`), representando o diálogo entre usuário e assistente. Um exemplo simplificado é apresentado a seguir:

```
{
  "model": "agnet-ministral",
  "messages": [
    {"role": "user", "content": "criar rede"},
    {"role": "assistant", "content": "((ip netns add h1))"},
    {"role": "user", "content": "Saida do comando...\n
    Quais os proximos passos?"}
  ]
}
```

Como o modelo não mantém estado entre chamadas, todo o contexto é explicitamente reenviado a cada requisição. O agente opera em um ciclo iterativo no qual cada resposta gerada é tratada como um comando, executada no sistema e, em seguida, seu resultado é reinserido no histórico. Dessa forma, o processo estabelece um fluxo contínuo de geração, execução e realimentação, possibilitando raciocínio incremental baseado no estado atual do ambiente e garantindo consistência ao longo da interação.

3.8.9. Aspectos de Segurança e Controle do Executor

Embora o modelo seja responsável por propor comandos, o controle efetivo da execução permanece no orquestrador Python. Essa arquitetura permite registrar todas as interações em arquivo de log, limitar o número máximo de ciclos, padronizar determinados comportamentos e, potencialmente, inserir filtros adicionais antes da execução dos comandos.

No protótipo atual, o executor já aplica uma normalização simples para comandos de `ping`, forçando o uso de `ping -c 3`. Esse mecanismo ilustra como políticas operacionais podem ser implementadas no executor para reforçar previsibilidade, repetibilidade e segurança.

Essa abordagem segue o princípio de *guardrails*, no qual o executor atua como uma camada de contenção, limitando o espaço de ações do agente e garantindo previsibilidade operacional.

Validação:

```
ip netns exec ns1 ping -c 2 10.0.0.2
```

A interpretação do resultado do `ping` permite ao agente confirmar a conectividade ou iniciar um processo de correção.

Controle do executor e normalização de comandos No executor implementado, comandos potencialmente contínuos, como o `ping`, são automaticamente normalizados para evitar execuções indefinidas. Por exemplo, qualquer chamada ao comando `ping` é transformada para incluir o parâmetro `-c 3`, limitando o número de pacotes enviados:

```
ping 10.0.0.2 → ping -c 3 10.0.0.2
```

Essa normalização é implementada diretamente no executor Python, conforme ilustrado a seguir:

```
cmd = re.sub(r'ping\s+(\d{1,3}(?:\.\d{1,3}){3})',
             r'ping -c 3 \1', cmd)
```

Esse mecanismo evita que o agente gere comandos que possam bloquear a execução (por exemplo, *ping* contínuo) e ilustra como políticas de segurança e controle podem ser incorporadas no executor.

De forma análoga, o executor pode ser estendido para filtrar ou bloquear comandos potencialmente destrutivos, reforçando a separação entre o processo de decisão do modelo e a execução controlada no sistema operacional.

Em conjunto, OpenWebUI, RAG, modelo clonado e executor Python compõem uma arquitetura de agente prático, na qual o modelo interpreta intenções e propõe ações, enquanto o executor observa, aplica e devolve o estado real do sistema para novo ciclo de decisão.

3.9. Segurança, Riscos e Boas Práticas no Uso de Agentes em Redes

Esta seção analisa os riscos críticos associados à implantação de agentes inteligentes em infraestruturas de rede e apresenta estratégias fundamentais para projetar sistemas seguros, auditáveis e resilientes.

3.9.1. Análise de Riscos

A transição de modelos de linguagem estáticos para agentes autônomos amplia significativamente a superfície de ataque, pois esses sistemas não apenas geram texto, mas também possuem a capacidade de invocar ferramentas e interagir diretamente com o ambiente digital e, em alguns casos, com sistemas físicos [Kong et al. 2025].

Execução arbitrária e uso indevido de ferramentas. Diferentemente de LLMs tradicionais, agentes podem produzir impactos concretos no ambiente operacional, como operar serviços de forma incorreta, corromper bases de dados ou comprometer sistemas críticos de rede [Kong et al. 2025]. A execução de código gerado automaticamente, como scripts em Python, representa um risco severo, pois pode envolver acesso não autorizado ao sistema de arquivos, comandos de *shell* maliciosos ou importações inseguras [Derouiche et al. 2025].

***Prompt injection*.** Esse é um dos riscos mais relevantes e pode ser dividido em duas categorias principais [Kong et al. 2025]:

- **Direta**: ocorre quando o usuário fornece instruções destinadas a sobrescrever o *system prompt* original, por exemplo, com comandos do tipo "ignore todas as instruções anteriores", buscando subverter o comportamento do agente [Kong et al. 2025].
- **Indireta**: ocorre quando o agente consome dados de fontes externas, como documentos recuperados via RAG ou páginas web, que contêm comandos maliciosos disfarçados. Nesse caso, o agente pode ser induzido a executar ações não planejadas, como vazar variáveis de ambiente ou segredos de configuração [Kong et al. 2025, Du et al. 2026].

Escalada de privilégio (*authority abuse*). Em sistemas multiagente, um invasor pode induzir um agente de menor privilégio a repassar um contexto manipulado para um agente com permissões superiores [Kong et al. 2025]. Caso o agente receptor herde essa confiança implicitamente, sem verificar o escopo original de autorização, o atacante poderá acionar operações administrativas fora de sua permissão [Kong et al. 2025].

Erros operacionais críticos. As alucinações técnicas podem levar à geração de configurações sintaticamente corretas, porém logicamente desastrosas, como uma ACL que isole o roteador de gerenciamento [Lira et al. 2024, Huang et al. 2023]. Além disso, ataques de exaustão cognitiva podem induzir o agente a entrar em ciclos excessivos de raciocínio e tentativa, consumindo recursos computacionais e degradando a disponibilidade do serviço de gerenciamento [Kong et al. 2025].

3.9.2. Estratégias de Mitigação e Boas Práticas

Para mitigar esses riscos, a arquitetura deve ser construída sob uma filosofia de defesa em profundidade, combinando controles sistêmicos, restrições operacionais e mecanismos especializados de segurança.

Princípios de *Zero Trust*. A segurança não deve depender de perímetros fixos, mas de verificação contínua. No contexto de agentes, isso implica autenticação mútua entre agentes e ferramentas, isolamento de contexto e verificação de privilégios em cada etapa do fluxo de trabalho [Kong et al. 2025, Boateng et al. 2024b]. Cada ferramenta deve ser executada com o princípio do menor privilégio, em ambientes isolados, como contêineres Docker com restrições rígidas de rede e sistema de arquivos [Derouiche et al. 2025, Bandara et al. 2025].

***Classifier models* e *guardrails*.** O uso de classificadores de segurança dedicados permite detectar tentativas de *jailbreak* e injeção de *prompt* antes que atinjam o núcleo cognitivo do sistema [OpenAI 2025]. Esses *guardrails* funcionam como uma camada defensiva que valida tanto a entrada do usuário quanto a saída do modelo, garantindo que as respostas permaneçam alinhadas às políticas de segurança e privacidade, incluindo filtros de dados sensíveis [OpenAI 2025].

***Alignment* e modelos de segurança.** Estratégias como a IA Constitucional buscam tornar os modelos mais seguros por meio de princípios explícitos de comportamento e regras operacionais, permitindo que o agente recuse solicitações maliciosas e justifique tecnica-

mente sua recusa [Bai et al. 2022]. Além disso, a adoção de um consórcio de modelos pode aumentar a robustez do sistema: múltiplos LLMs produzem saídas independentes, posteriormente validadas por um agente verificador, reduzindo a probabilidade de que uma única alucinação ou viés seja propagado [Bandara et al. 2025, Du et al. 2026].

Auditoria e limites operacionais. A governança do sistema requer mecanismos claros de observabilidade e contenção:

- **Logs de auditoria**: é essencial manter registros imutáveis de entradas, saídas, chamadas de ferramentas e rastros de execução observáveis, a fim de viabilizar análises forenses e detecção de anomalias [Park et al. 2023, Kong et al. 2025].
- **Limites de execução**: devem ser estabelecidos limiares rígidos para falhas, tempo de execução e número de turnos. Se um agente exceder o número permitido de tentativas de configuração ou tentar realizar uma ação de alto risco, como desligar uma interface principal, o sistema deve pausar a operação e exigir intervenção humana (*Human-in-the-Loop*) [OpenAI 2025, Lira et al. 2024].
- **Separação de lógica**: funções puramente operacionais, como escrita em disco, *commits* ou alterações persistentes de estado, devem ser delegadas a funções determinísticas implementadas em código, por exemplo em Python, reservando o LLM apenas para etapas que exijam interpretação semântica e raciocínio linguístico [Bandara et al. 2025].

Ao integrar essas práticas, operadores de rede podem construir um ecossistema de NetOps 2.0 em que a automação inteligente seja equilibrada por mecanismos rigorosos de governança, segurança e controle [Lira et al. 2024, Bandara et al. 2025].

3.10. Conclusão e Perspectivas Futuras

Esta seção final consolida os principais aprendizados deste minicurso e discute perspectivas futuras para o gerenciamento de infraestruturas de rede, em um cenário no qual a convergência entre modelos de linguagem, agentes inteligentes e protocolos de rede redefine a autonomia operacional.

A trajetória do NetOps 2.0 representa a transição de sistemas baseados em scripts estáticos para ecossistemas de inteligência de rede unificada [Huang et al. 2023]. Nesse cenário, a IA generativa deixa de atuar apenas como assistente textual e passa a ocupar o papel de núcleo cognitivo de sistemas capazes de perceber, planejar e agir sobre a infraestrutura de forma autônoma ou semiautônoma [Kong et al. 2025, Elkael et al. 2026].

3.10.1. Consolidação da Arquitetura Multimodelo

O paradigma contemporâneo abandonou a dependência de modelos únicos e monolíticos em favor de fluxos de trabalho agênticos (*Agentic AI Workflows*) [Bandara et al. 2025]. Nessa arquitetura modular, responsabilidades são distribuídas entre componentes especializados, cada qual com papel definido no ciclo de operação:

- **Orquestradores e roteadores**: responsáveis por gerenciar o fluxo de tarefas e selecionar os recursos mais adequados a cada demanda [de Lamo Castrillo et al. 2025, Bandara et al. 2025].

- **Modelos de raciocínio e verificadores**: encarregados de sustentar consistência, segurança e confiabilidade por meio de planejamento, autorreflexão e validação sintática ou semântica [de Lamo Castrillo et al. 2025, Lira et al. 2024].
- **Modelos locais (SLMs)**: fundamentais para assegurar privacidade de dados e baixa latência em operações críticas executadas na borda (*edge*) ou em ambientes com maior restrição operacional [Lira et al. 2024].

3.10.2. O Papel Transformador da IA Generativa em Redes

LLMs e VLMs (*Vision-Language Models*) ampliaram significativamente a capacidade de processar dados não estruturados e multimodais, como telemetria, diagramas de topologia e logs complexos, permitindo uma compreensão mais ampla do estado da rede [de Lamo Castrillo et al. 2025, Zhou et al. 2024]. A principal inovação reside na interpretação de intenção (*intent-driven*), que reduz a distância entre objetivos de negócio e execução técnica, viabilizando que a rede se autoconfigure a partir de comandos expressos em linguagem natural [Huang et al. 2023, Lira et al. 2024].

3.10.3. Tendências e Desdobramentos para o Horizonte 2026–2030

A evolução da automação de redes no horizonte de 2026 a 2030 será marcada pela convergência entre inteligência distribuída, arquiteturas orientadas a agentes e maior autonomia operacional. Nesse cenário, observa-se a transição de sistemas reativos e centralizados para ecossistemas capazes de perceber, decidir e agir de forma contínua, com mínima intervenção humana.

3.10.3.1. Tendências Estruturais

As principais tendências estruturais apontam para uma redefinição do paradigma de operação de redes, com destaque para os seguintes eixos:

- **Redes autônomas e ZSM**: a consolidação do *Zero-touch Network & Service Management* (ZSM) amplia a capacidade de auto-operação, autoconfiguração e automanutenção das redes. Nesse modelo, a atuação humana desloca-se para funções de supervisão, definição de políticas e governança, enquanto os sistemas executam ciclos fechados de controle [Lira et al. 2024, Elkael et al. 2026].
- **Sistemas multiagentes e Internet of Agents (IoA)**: a emergência de ecossistemas compostos por múltiplos agentes inteligentes apresenta novos desafios. Nesse contexto, entidades autônomas são capazes de descobrir serviços, negociar estratégias e cooperar dinamicamente por meio de protocolos específicos, como MCP e A2A [Kong et al. 2025, Elkael et al. 2026].
- **Integração com OpenRAN**: a incorporação de inteligência agentica ambientes OpenRAN, especialmente com as r/xApps, viabiliza o controle em malha fechada com granularidade temporal reduzida, aproximando a automação de requisitos operacionais em tempo real o que é essencial para diversas aplicações e serviços do ecossistema OpenRAN [Elkael et al. 2026].
- **Redes autoadaptativas e *self-healing***: a evolução para redes capazes de aprender

continuamente a partir de dados operacionais permite a identificação proativa de falhas, degradações e interferências, com geração automática de estratégias de mitigação. Iniciativas como AI-RAN Factory ilustram esse movimento em direção a sistemas autoevolutivos [Elkael et al. 2026, Boateng et al. 2024b].

3.10.3.2. Desdobramentos Práticos e Tecnológicos

A partir dessas tendências estruturais, emergem desdobramentos concretos que impactam diretamente a implementação e operação de sistemas de rede:

- **Arquiteturas *SLM-first***: observa-se a consolidação de arquiteturas heterogêneas nas quais *Small Language Models* (SLMs) atuam como padrão operacional para tarefas recorrentes, enquanto *Large Language Models* (LLMs) são acionados sob demanda para cenários que exigem maior capacidade de raciocínio. O ajuste fino de SLMs sobre dados organizacionais, como logs, *playbooks* e históricos de configuração, tende a proporcionar ganhos significativos em eficiência e custo [Microsoft Research 2025, Liquid AI 2025].
- **Interoperabilidade entre agentes**: a evolução de protocolos como MCP e A2A em direção a modelos de governança abertos favorece a interoperabilidade entre agentes e sistemas heterogêneos. Espera-se que controladores SDN, plataformas de monitoramento e orquestradores de NFV passem a expor interfaces nativas compatíveis, reduzindo a necessidade de integrações específicas e promovendo maior composição de serviços [Anthropic 2025, Google 2025].
- **Consolidação da *Edge AI***: a viabilidade de execução de modelos compactos em dispositivos com recursos limitados amplia o uso de agentes inteligentes em ambientes de borda, como IoT industrial e infraestruturas O-RAN. A inferência local passa a desempenhar papel central não apenas na redução de latência e custo, mas também na garantia de disponibilidade em cenários com conectividade intermitente.
- **Novos paradigmas de avaliação**: a avaliação de sistemas inteligentes desloca-se do desempenho isolado de modelos para a análise de sistemas completos capazes de tomar decisões e executar ações. *Benchmarks* recentes, como SWE-bench Verified e GAIA, evidenciam a importância de métricas relacionadas à qualidade do raciocínio intermediário, eficiência de planejamento e capacidade de autocorreção [SWE-bench Team 2025, Mialon et al. 2023].

3.10.4. Oportunidades de Pesquisa e Inovação

Apesar dos avanços recentes, diversas áreas ainda demandam investigação técnica e acadêmica aprofundada:

- **Interoperabilidade entre domínios**: desenvolvimento de arquiteturas de agentes capazes de operar de forma integrada em ambientes heterogêneos, incluindo segmentos satelitais, aéreos e terrestres [Boateng et al. 2024b, Du et al. 2026].
- **Eficiência de inferência em tempo real**: otimização de modelos e arquiteturas para atender aos requisitos de latência extremamente baixa previstos para redes 6G e sistemas críticos distribuídos [Boateng et al. 2024b, Chowa et al. 2026].

- **Segurança e responsabilização**: criação de mecanismos de auditoria forense, rastreabilidade de decisão e atribuição de responsabilidade em falhas causadas por agentes, além de proteção contra ataques de exaustão cognitiva e abuso de ferramentas [Kong et al. 2025, Bandara et al. 2025].
- **Padronização de *benchmarks***: estabelecimento de métricas robustas para avaliar inteligência, confiabilidade, custo operacional e segurança de agentes em cenários de rede realistas [Du et al. 2026, Chowa et al. 2026].

Em síntese, este minicurso demonstrou que a construção de agentes inteligentes constitui um passo decisivo para transformar redes em sistemas mais resilientes, adaptativos e capazes de ampliar continuamente sua inteligência operacional. Mais do que automatizar tarefas isoladas, a combinação entre modelos de linguagem, arquiteturas de agentes, RAG e mecanismos de verificação aponta para uma nova geração de infraestruturas capazes de aprender com a própria operação e responder de forma mais eficaz às demandas da próxima década [Elkael et al. 2026, Huang et al. 2023].

Agradecimentos: A elaboração deste minicurso faz parte de projetos apoiados pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248/1991, no âmbito do PPI-SOFTEX, coordenado pela Softex e publicado na Arquitetura Cognitiva (Fase 3), DOU 01245.003479/2024-10, bem como de projetos do INCT Redes de Comunicação e Internet das Coisas Inteligentes, financiados pelo CNPq (processo nº 405940/2022-0) e FAPESP (processo nº 2023/00673-7).

Referências

[Anthropic 2024] Anthropic (2024). Contextual retrieval. `https://www.anthropic.com/news/contextual-retrieval`.

[Anthropic 2025] Anthropic (2025). Model context protocol specification. modelcontextprotocol.io.

[Bai et al. 2022] Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. (2022). Constitutional ai: Harmlessness from ai feedback.

[Bandara et al. 2025] Bandara, E., Gore, R., Foytik, P., Shetty, S., Mukkamala, R., Rahman, A., Liang, X., Bouk, S. H., Hass, A., Rajapakse, S., Keong, N. W., Zoysa, K. D., Withanage, A., and Loganathan, N. (2025). A practical guide for designing, developing, and deploying production-grade agentic ai workflows.

[Boateng et al. 2024a] Boateng, G. O. et al. (2024a). A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions. *arXiv preprint arXiv:2412.19823*.

[Boateng et al. 2024b] Boateng, G. O., Sami, H., Alagha, A., Elmekki, H., Hammoud, A., Mizouni, R., Mourad, A., Otrok, H., Bentahar, J., Muhaidat, S., Talhi, C., Dziong, Z., and Guizani, M. (2024b). A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions.

[Bommasani et al. 2022] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. (2022). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[Borgeaud et al. 2022] Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J.-B., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J. W., Elsen, E., and Sifre, L. (2022). Improving language models by retrieving from trillions of tokens.

[Chakraborty et al. 2024] Chakraborty, S., Chitta, N., and Sundaresan, R. (2024). Automation of network configuration generation using large language models. In *Proc. 20th International Conference on Network and Service Management (CNSM)*.

[Chowa et al. 2026] Chowa, S. S., Alvi, R., Rahman, S. S., Rahman, M. A., Raiaan, M. A. K., Islam, M. R., Hussain, M., and Azam, S. (2026). From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 59(2).

[de Lamo Castrillo et al. 2025] de Lamo Castrillo, V., Gidey, H. K., Lenz, A., and Knoll, A. (2025). Fundamentals of building autonomous llm agents.

[Derouiche et al. 2025] Derouiche, H., Brahmi, Z., and Mazeni, H. (2025). Agentic AI frameworks: Architectures, protocols, and design challenges. arXiv:2508.10146.

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

[Du et al. 2022] Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q. V., Wu, Y., Chen, Z., and Cui, C. (2022). Glam: Efficient scaling of language models with mixture-of-experts.

[Du et al. 2026] Du, S., Zhao, J., Shi, J., Xie, Z., Jiang, X., Bai, Y., and He, L. (2026). A survey on the optimization of large language model-based agents. *ACM Comput. Surv.*, 58(9).

[Elkael et al. 2026] Elkael, M., D'Oro, S., Bonati, L., Polese, M., Lee, Y., Furueda, K., and Melodia, T. (2026). Agentran: An agentic ai architecture for autonomous control of open 6g networks.

[Erdogan et al. 2024] Erdogan, L. E., Lee, N., Jha, S., Kim, S., Tabrizi, R., Moon, S., Hooper, C., Anumanchipalli, G., Keutzer, K., and Gholami, A. (2024). Tinyagent: Function calling at the edge.

[Errico et al. 2025] Errico, H., Ngiam, J., and Sojan, S. (2025). Securing the model context protocol (mcp): Risks, controls, and governance.

[Fedus et al. 2022] Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv:2101.03961.

[Gao et al. 2023] Gao, Y. et al. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

[Google 2025] Google (2025). Agent-to-agent (A2A) protocol specification. Technical report, Google.

[Ha and Schmidhuber 2018] Ha, D. and Schmidhuber, J. (2018). World models. Zenodo. DOI: 10.5281/zenodo.1207631.

[Huang et al. 2023] Huang, Y., Du, H., Zhang, X., Niyato, D., Kang, J., Xiong, Z., Wang, S., and Huang, T. (2023). Large language models for networking: Applications, enabling techniques, and challenges.

[Jano 2024] Jano, P. W. (2024). Retrieval-augmented generation systems: A comprehensive survey of architectures, applications, and future directions. *University of Wisconsin–Madison Survey*.

[Jhandi et al. 2026] Jhandi, P., Kazi, O., Subramanian, S., and Sendas, N. (2026). Small language models for efficient agentic tool calling: Outperforming large models with targeted fine-tuning.

[Kaplan et al. 2020] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. arXiv:2001.08361.

[Karpukhin et al. 2020] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. (2020). Dense passage retrieval for open-domain question answering. arXiv:2004.04906.

[Khattab and Zaharia 2020] Khattab, O. and Zaharia, M. (2020). ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. arXiv:2004.12832.

[Kim 2026] Kim, M. H. (2026). Emergent cognitive convergence via implementation: Structured cognitive loop reflecting four theories of mind.

[Kong et al. 2025] Kong, D., Lin, S., Xu, Z., Wang, Z., Li, M., Li, Y., Zhang, Y., Peng, H., Chen, X., Sha, Z., Li, Y., Lin, C., Wang, X., Liu, X., Zhang, N., Chen, C., Wu, C., Khan, M. K., and Han, M. (2025). A survey of llm-driven ai agent communication: Protocols, security risks, and defense countermeasures.

[LangChain 2025] LangChain (2025). LangGraph: Building stateful multi-actor applications with LLMs. github.com/langchain-ai/langgraph.

[Lewis et al. 2020] Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*.

[Lewis et al. 2021] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2021). Retrieval-augmented generation for knowledge-intensive nlp tasks.

[Liquid AI 2025] Liquid AI (2025). LFM2.5: Liquid foundation models technical report. Technical report, Liquid AI.

[Lira et al. 2024] Lira, O. G., Caicedo, O. M., and da Fonseca, N. L. (2024). Large language models for zero touch network configuration management. *IEEE Communications Magazine*, 63(7):146–153.

[Liu et al. 2025] Liu, J., Kong, Z., Yang, C., Yang, F., Li, T., Dong, P., Nanjekye, J., Tang, H., Yuan, G., Niu, W., Zhang, W., Zhao, P., Lin, X., Huang, D., and Wang, Y. (2025). Rcr-router: Efficient role-aware context routing for multi-agent llm systems with structured memory.

[Long et al. 2025] Long, S., Tan, J., Mao, B., Tang, F., Li, Y., Zhao, M., and Kato, N. (2025). A survey on intelligent network operations and performance optimization based on large language models. *IEEE Communications Surveys & Tutorials*, 27(6):3915–3949.

[Madaan et al. 2023] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. (2023). Self-refine: Iterative refinement with self-feedback.

[Manning et al. 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Mialon et al. 2023] Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. (2023). Gaia: a benchmark for general ai assistants.

[Microsoft Research 2025] Microsoft Research (2025). Phi-4-mini technical report. Technical report, Microsoft.

[ML.ENERGY 2024] ML.ENERGY (2024). ML.ENERGY leaderboard: Energy consumption of large language models. ml.energy/leaderboard.

[NVIDIA Research 2025] NVIDIA Research (2025). Small language models are the future of agentic AI. arXiv:2510.03847.

[OpenAI 2025] OpenAI (2025). A practical guide to building agents. Technical report, OpenAI.

[Ouyang et al. 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.

[Pan et al. 2024] Pan, J. J., Wang, J., and Li, G. (2024). Survey of vector database management systems. *The VLDB Journal*.

[Park et al. 2023] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior.

[Rawat et al. 2025] Rawat, M., Gupta, A., Goomer, R., Bari, A. D., Gupta, N., and Pieraccini, R. (2025). Pre-act: Multi-step planning and reasoning improves acting in llm agents.

[Robertson and Zaragoza 2009] Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*.

[Sanh et al. 2020] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

[Schick et al. 2023] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools.

[Sharma and Mehta 2025] Sharma, R. and Mehta, M. (2025). Small language models for agentic systems: A survey of architectures, capabilities, and deployment trade offs.

[Singh et al. 2025] Singh, A., Ehtesham, A., Kumar, S., and Khoei, T. T. (2025). Agentic retrieval-augmented generation: A survey on agentic rag.

[SWE-bench Team 2025] SWE-bench Team (2025). SWE-bench verified leaderboard. swebench.com.

[Webb et al. 2025] Webb, T., Mondal, S. S., and Momennejad, I. (2025). Improving planning with large language models: A modular agentic architecture.

[Yao et al. 2023] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). React: Synergizing reasoning and acting in language models.

[Ye et al. 2025] Ye, R., Liu, X., Wu, Q., Pang, X., Yin, Z., Bai, L., and Chen, S. (2025). X-mas: Towards building multi-agent systems with heterogeneous llms.

[Yue et al. 2025] Yue, Y., Zhang, G., Liu, B., Wan, G., Wang, K., Cheng, D., and Qi, Y. (2025). Masrouter: Learning to route llms for multi-agent systems.

[Zhou et al. 2024] Zhou, H., Hu, C., Yuan, Y., Cui, Y., Jin, Y., Chen, C., Wu, H., Yuan, D., Jiang, L., Wu, D., Liu, X., Zhang, C., Wang, X., and Liu, J. (2024). Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities.

Capítulo

4

Gestão de Chaves em Redes Quantum-Safe: QKD, KMS e Integração com Sistemas Clássicos

Adriano Maia (UFBA/QuIIN), Isys Sant'Anna (UFBA/QuIIN), Marcus Freire (UFBA/QuIIN), Thiago Mello (UFBA/QuIIN), Anderson Tomkelski (QuIIN), Gabriel Caldas (UFBA), João Souza (QuIIN), Ricardo Parizotto (UFFS), Bruno Santos (UFBA) e Maycon Peixoto (UFBA)

Abstract

As quantum communications advance, the standardization of key management has become a fundamental pillar for achieving global interoperability. This chapter analyzes the regulatory frameworks of ETSI ISG QKD and the ITU-T Y.3800 series, providing in-depth insights into the functional architecture, requirements, and operational procedures of Quantum Key Distribution Networks (QKDN). The study highlights the differences and complementarities between key delivery interfaces (REST API) and network infrastructure management. Furthermore, the chapter includes a practical section dedicated to simulating real-world scenarios using GNS3, demonstrating the configuration of IPsec security integrated with QKD systems. The chapter concludes with a synthesis of technological challenges and future trends in hybrid quantum security.

Resumo

Com o avanço das comunicações quânticas, a padronização do gerenciamento de chaves tornou-se um pilar fundamental para a interoperabilidade global. Este capítulo analisa os frameworks normativos do ETSI ISG QKD e da série ITU-T Y.3800, fornecendo uma visão ampla sobre a arquitetura funcional, requisitos e procedimentos operacionais das Redes de Distribuição de Chaves Quânticas (QKDN). O estudo destaca as diferenças e complementaridades entre as interfaces de entrega de chaves (REST API) e a gestão da infraestrutura de rede. A obra inclui ainda uma seção prática dedicada à simulação de cenários reais via GNS3, onde é demonstrada a configuração de segurança IPsec integrada a sistemas QKD. O capítulo encerra com uma síntese dos desafios tecnológicos e as tendências futuras para a segurança quântica híbrida.

4.1. Introdução

O avanço da computação quântica impõe novos desafios à segurança da informação, ao ameaçar sistemas criptográficos convencionais baseados em problemas matemáticos de difícil solução, como a fatoração de inteiros e o logaritmo discreto, tradicionalmente considerados intratáveis para a computação clássica [Ahmed et al. 2025, Easttom 2022, Nielsen and Chuang 2010]. Esse cenário evidencia a necessidade de mecanismos de proteção que não se fundamentem apenas em hipóteses de dificuldade computacional. Nesse sentido, a Distribuição Quântica de Chaves (do Inglês *Quantum Key Distribution (QKD)*) destaca-se como uma alternativa promissora para a segurança das comunicações, por basear sua proteção nas leis da mecânica quântica. Essa abordagem possibilita a geração e o compartilhamento de chaves criptográficas simétricas com segurança teoricamente incondicional, constituindo uma estratégia relevante para a proteção de dados sensíveis em cenários futuros marcados pela computação quântica [Gisin et al. 2002].

Embora a QKD ofereça garantias de segurança fundamentadas nas leis da física [Narroway 2025], sua aplicação em ambientes reais ainda esbarra em limitações tecnológicas e operacionais significativas. Entre os principais entraves, destacam-se as perdas em canais ópticos, que impõem restrições ao alcance da comunicação, as limitações na taxa de geração de chaves, a necessidade de canais clássicos autenticados e a dependência de nós confiáveis para viabilizar comunicações em longas distâncias. Além desses aspectos, sua integração com as infraestruturas de telecomunicações existentes demanda mecanismos eficientes de gerenciamento, armazenamento, sincronização e distribuição do material criptográfico. Como consequência, a QKD configura-se, essencialmente, como uma tecnologia de enlace ponto a ponto, o que exige arquiteturas de rede e soluções de gestão de chaves capazes de sustentar sua escalabilidade e favorecer sua adoção em larga escala [Dervisevic et al. 2025, Mehic et al. 2020].

Diante dessas demandas, este minicurso adota uma abordagem orientada à engenharia de redes e sistemas, com ênfase na gestão do ciclo de vida das chaves criptográficas e em sua integração com infraestruturas de segurança já consolidadas. Nessa perspectiva, são discutidas arquiteturas e iniciativas de padronização internacional, com destaque para as recomendações do *European Telecommunications Standards Institute (ETSI)* [ETSI 2019] e da *International Telecommunication Union – Telecommunication Standardization Sector (ITU-T)* [ITU-T 2019], bem como para os conceitos de *Quantum Key Distribution Networks (QKDN)* e *Key Management Systems (KMS)*. Tal abordagem permite compreender de que maneira chaves criptográficas geradas por sistemas QKD podem ser incorporadas ao plano de controle da segurança e empregadas de forma integrada a protocolos clássicos, como o IPsec.

Além da fundamentação teórica, este capítulo adota uma abordagem teórico-prática. Neste sentido, propomos uma interação em laboratório prático usando uma API REST baseada na especificação *ETSI Group Specification (ETSI GS)* QKD 014. Nesta atividade, você pode assumir o papel de *Secure Application Entitiess (SAEs)*, isto é, entidades de aplicação que consomem chaves criptográficas fornecidas pela infraestrutura de gerenciamento de chaves, abrangendo, na prática, o fluxo de solicitação, entrega e sincronização de chaves em ambientes híbridos clássico-quânticos. O laboratório inclui a implementação de uma *Key Management Entity (KME)* e a integração com uma VPN

IPsec em ambiente simulado, evidenciando a aplicação dos padrões internacionais na proteção de infraestruturas de rede. Dessa forma, pretende-se contribuir para a capacitação de estudantes, pesquisadores e profissionais no projeto, na implementação e na integração de soluções de segurança *Quantum-Safe*, promovendo o avanço das redes seguras de próxima geração.

Este capítulo está estruturado de maneira progressiva, da fundamentação teórica à experimentação prática de redes QKDN. Após a apresentação conceitual dos fundamentos do QKD e da necessidade estrutural do KMS, serão discutidos os conceitos de arquitetura e padronização, detalhando o modelo de referência da ITU-T e, logo após, a arquitetura e as interfaces de programação estabelecidas pela ETSI. Compreendida essa base de comunicação e interface, o texto aprofundará as questões de segurança e confiança do KMS, aspecto que garante a proteção e a integridade da infraestrutura. Para ilustrar os conceitos expostos, a teoria será conectada à prática por meio de uma simulação de cenário onde a segurança de uma VPN IPsec é reforçada por chaves quânticas em operação conforme com o padrão ETSI GS QKD 014, culminando em uma conclusão que aborda os desafios e o futuro da área. Essa organização foi adotada com o objetivo de explorar a essência do ecossistema QKDN, desde os fundamentos físicos até as interfaces de rede, possibilitando ao leitor condições de explorar com maior facilidade a implementação real e as futuras evoluções dessa tecnologia.

4.2. Do QKD ao Problema da Gestão de Chaves

A segurança das comunicações digitais modernas fundamenta-se historicamente no emprego de técnicas de criptografia simétrica e assimétrica [Mehic et al. 2020, Stallings 2017]. No escopo da criptografia simétrica, os processos de cifragem e decifragem exigem que a mesma chave seja mantida em estrito sigilo entre as partes comunicantes. Por outro lado, no paradigma de criptografia assimétrica (ou de chave pública), cada entidade gera um par de chaves correlacionadas matematicamente: uma pública, distribuída abertamente para o processo de cifragem, e uma privada, mantida em sigilo absoluto para a decifragem. Na prática, a criptografia simétrica é mais utilizada para cifragem, enquanto a assimétrica é utilizada para distribuir chaves para sistemas criptográficos simétricos [Gisin et al. 2002, Stallings 2017].

A resiliência da criptografia assimétrica, entretanto, baseia-se em premissas de complexidade computacional. Algoritmos amplamente adotados na proteção da Internet, como o Rivest-Shamir-Adleman (RSA) e a Elliptic-Curve Cryptography (ECC), fundamentam sua inviolabilidade na dificuldade prática de resolver problemas matemáticos de via única, notadamente a fatoração de grandes inteiros e o cálculo de logaritmos discretos. A validade desta abordagem, sob o ponto de vista da segurança prática que ela é capaz de proporcionar, baseia-se na hipótese de que a reversão de tais operações é intratável computacionalmente para a arquitetura dos processadores clássicos [Stallings 2017].

É precisamente a dependência da complexidade computacional que representa a principal ameaça contemporânea à segurança das redes. O avanço no desenvolvimento de computadores quânticos tolerantes a falhas desestabiliza essa premissa, uma vez que tais máquinas serão capazes de solucionar problemas de fatoração e de logaritmo discreto em tempo polinomial por meio da execução de algoritmos como o de

Shor [Sanz et al. 2025, Lai et al. 2023, Bhatia and Ramkumar 2020]. Por conseguinte, a infraestrutura assimétrica torna-se estruturalmente vulnerável, exigindo que a segurança de redes passe a adotar modelos imunes a esse novo vetor de ataque.

Em resposta a esse cenário, a distribuição de chaves quânticas consolida-se como uma alternativa eficaz para o estabelecimento seguro de material criptográfico [Lai et al. 2023]. Diferentemente dos criptossistemas clássicos, a QKD não pauta sua segurança pela complexidade matemática, mas sim pelas leis fundamentais da física [Elboukhari et al. 2010]. A imunidade do protocolo contra interceptações é assegurada pelo princípio da incerteza de Heisenberg e pelo teorema da não-clonagem [Sanz et al. 2025, Gisin et al. 2002]. Qualquer tentativa de medição não autorizada no canal quântico perturbará irreversivelmente o estado não ortogonal dos fótons transmitidos, induzindo um aumento na Taxa de Erro de Bits Quânticos (*Quantum Bits Error Rate (QBER)*), que será detectado pelas partes legítimas. Desta forma, a QKD extingue a incerteza probabilística da espionagem computacional quando avalia-se o ajuste do canal ao limite estatístico tolerável para a perda de bits quânticos (*qubit*) e torna-se possível classificar se a QBER corresponde a um provável produto das interferências eletromagnéticas naturais à transmissão dos fótons em canais físicos, ou se corresponde a uma provável espionagem [Lai et al. 2023].

4.2.1. O que o QKD resolve (e o que não resolve)

Sob a perspectiva da arquitetura de uma rede baseada em QKD, esta opera na chamada camada quântica [ITU-T 2019]. Contudo, a natureza do seu funcionamento permite classificá-la, por analogia ao modelo *Open System Interconnection (OSI)* [Day and Zimmermann 1984], como um protocolo de camadas física e de enlace, com a função específica de atuar como um mecanismo provável e fisicamente seguro para o estabelecimento de chaves [ITU-T 2019, Mehic et al. 2020]. O problema central que essa tecnologia soluciona é a distribuição de chaves por um canal interceptável.

A operação completa exige a coexistência de um canal quântico, responsável pela transmissão física dos qubits, e de um canal clássico, no qual ocorrem as fases análogas à camada clássica de enlace [ITU-T 2019]. A etapa de natureza física é a transmissão de fótons em um canal quântico de fibra óptica, transmissão essa que tem o seu insucesso intimamente associado às imperfeições do meio de propagação e às consequentes interferências eletromagnéticas que degradam os dados transmitidos e gera aumento na QBER [Bennett and Brassard 2014].

Para recuperar a integridade da informação, ainda com a parcela de dados corrompidos no processo físico, a processo de QKD demanda o uso do canal clássico em operação análoga à camada de enlace do modelo OSI [Mehic et al. 2020]. A QKD utiliza o canal público clássico para processar os dados brutos recebidos [ITU-T 2019, Gisin et al. 2002]. Essa fase de deputação engloba a reconciliação de bases incompatíveis (*sifting*), correção algorítmica de erros e amplificação de privacidade. Assim, consolida-se uma chave simétrica que pode ser classificada como segura, dada a impossibilidade física de espionagem não detectável, pronta para alimentar cifras como o *Advanced Encryption Standard (AES)* ou o *One-Time Pad (OTP)* [Gisin et al. 2002].

Contudo, é destacável a fragilidade da proposta perante a autenticação, visto

que o canal quântico garante o sigilo contra interceptações passivas, mas não dispõe nativamente de mecanismos que validem a identidade das partes em comunicação. A fase de reconciliação no canal clássico é vulnerável a ataques do tipo Homem no Meio (*Man-in-the-Middle (MitM)*) [Elboukhari et al. 2010], em que um adversário poderia se passar pelos usuários legítimos e estabelecer chaves independentes de origem e destino [Gisin et al. 2002]. Portanto, os protocolos de QKD requerem que as extremidades no canal clássico sejam autenticadas [Bennett and Brassard 2014]. Vale destacar que pelo produto do processo de distribuição das chaves, a adoção da QKD reintroduz a utilização de criptografia simétrica e as suas respectivas limitações de eficiência operacional.

Além das fragilidades na segurança das identidades e da ineficiência na gestão de chaves geradas, a QKD não fornece uma estrutura metodológica suficiente para um cenário prático de redes. Não há, por definição, a gestão do ciclo de vida das chaves geradas; não são impostas as políticas de acesso e as chaves não são formatadas nativamente para os padrões exigidos por protocolos como Internet Protocol Securiry (IPsec) ou *Transport Layer Security (TLS)* [ITU-T 2019, Gisin et al. 2002]. Somam-se a isso as limitações topológicas impostas pelo teorema da não-clonagem [Bennett and Brassard 2014], que impede a regeneração do sinal por amplificadores ópticos clássicos. A atenuação natural da fibra restringe o alcance a conexões ponto a ponto de distâncias limitadas [Mehic et al. 2020, Lai et al. 2023]. Escalar a rede para interligar múltiplos usuários exigiria uma infraestrutura de malha física completa, atualmente impraticável. A soma destas limitações evidencia que a QKD, isoladamente, é computacionalmente estéril para a prestação de serviços fim a fim, exigindo, obrigatoriamente, a sobreposição de uma infraestrutura clássica de orquestração.

4.2.2. Por que o KMS é necessário em sistemas QKD

A tecnologia QKD atua, por definição, com estruturas de enlaces ponto a ponto [Lai et al. 2023]. A expansão da escala de aplicação da tecnologia em redes de múltiplos nós é viabilizada pela arquitetura padronizada em um modelo de camadas lógicas independentes, as *Quantum Key Distribution Networks* [ITU-T 2019]. Essa arquitetura divide o sistema fundamentalmente em três camadas: a camada quântica, a camada de gerenciamento de chaves e as camadas de controle e gerenciamento [ITU-T 2019]. Embora a QKD, atuante na camada quântica, resolva o desafio da distribuição de chaves criptográficas seguras, esta tecnologia atua em níveis "distantes" das camadas de aplicação na rede do usuário [Mehic et al. 2020].

As aplicações nas redes dos usuários, sejam túneis IPsec, *Media Access Control Security (MACsec)* ou sistemas de criptografia simétrica, exigem chaves devidamente formatadas, exigência essa a que o fluxo de bits aleatórios brutos gerado via QKD não consegue atender de forma direta [Dervisevic et al. 2025]. O Sistema de Gerenciamento de Chaves é o sistema que atua na camada de gerenciamento de chaves de uma QKDN, responsável por entregar as chaves prontas às aplicações de rede do usuário, a partir do produto da camada quântica. Sua arquitetura, interfaces e modelos de operação serão detalhados na Seção 4.3.1.

A camada quântica fornece ao KMS cadeias de bits que são redimensionadas por meio de fatiamento ou agrupamento em blocos de tamanhos fixos exigidos pelas apli-

cações criptográficas da rede do usuário [Mehic et al. 2020]. Uma vez que os bits estão dimensionados no tamanho adequado, o KMS formata os blocos anexando-lhes cabeçalhos (*headers*) e rodapés (*footers*) lógicos com metadados úteis para a sincronização e rastreamento na rede [ITU-T 2020a]. Após a estruturação dos metadados, a cadeia de bits formatada é convertida usando codificadores padrão, como o *Base64* e envelopada em estruturas de dados legíveis pelas aplicações da rede clássica, como o formato *JavaScript Object Notation (JSON)* [ETSI 2019]. Por fim, a partir de um canal clássico de comunicação dedicado, o sistema comunica com o KMS vizinho na rede para sincronizar e autenticar as chaves formatadas, garantindo que o armazenamento de ambos os lados em comunicação possua o mesmo material redimensionado para que a chave possa estar disponível para consumo [ITU-T 2019].

Além da adaptação estrutural dos dados, a necessidade do KMS em redes quânticas fundamenta-se na superação das limitações inerentes às fragilidades das partículas quânticas em trânsito na camada física [Dervisevic et al. 2025]. O tratamento da instabilidade na geração de bits e a restrição de distância de transmissão de partículas são os pontos que sustentam a importância do sistema em uma QKDN.

Por definição, a taxa com que o meio físico de transmissão de *qubits* gera chaves é inconstante [Mehic et al. 2020]. Para evitar que as aplicação fiquem sem chaves criptográficas em momentos de execução, o KMS funciona como um estoque de chaves prontas para uso, privando o serviço de uma possível indisponibilidade por conta das oscilações físicas do hardware associado à QKD [ITU-T 2019].

O KMS também se mostra útil na superação de outro obstáculo na aplicação prática de QKDNs, que é a limitação da distância de fibra óptica, de no máximo 100 quilômetros [Mehic et al. 2020, Narroway 2025]. O KMS, propõe o tratamento desse gargalo a partir do revezamento de chaves (*key relay*) [ITU-T 2019]. Utilizando uma rede de nós intermediários, os sistemas KMS encaminham a chave nó a nó até o destino final [ITU-T 2019, Mehic et al. 2020], criptografando-a a cada salto entre nós. Desta forma, o KMS permite que aplicações em extremidades fisicamente distantes de uma rede compartilhem chaves simétricas mesmo sem uma conexão óptica direta, o que viabiliza, praticamente, uma estrutura QKDN [ITU 2023].

4.2.3. Visão geral do ciclo de vida de uma chave

A operação de uma rede QKDN e sua integração à camada de gerenciamento ocorrem ao longo do ciclo de vida do seu material criptográfico. Esse ciclo abrange a sequência de estágios que uma cadeia de bits percorre desde a sua geração física, passando pelo suprimento a uma aplicação na camada de serviço, até a sua exclusão [ITU-T 2019]. Em arquiteturas baseadas em nós confiáveis, o ciclo de vida da chave desenrola-se em um fluxo contínuo.

A geração da chave ocorre na camada quântica, onde os módulos transmissores (QKD-Tx) e receptores (QKD-Rx) executam protocolos físicos, como o BB84, ou protocolos de estados coerentes, e o processamento ocorre no canal clássico [Gisin et al. 2002]. O fluxo de bits resultante é transferido para a camada de gerenciamento, ingressando em repositórios lógicos temporários do KMS local, denominados *pickup stores* [Mehic et al. 2020, ITU-T 2019]. Para garantir a consistência, os gerenciadores de

nós adjacentes interagem por meio de canais clássicos autenticados para confirmar a exatidão e a paridade das sequências geradas. Validadas, as chaves recebem metadados para o rastreamento lógico, como identificadores únicos e *timestamps* e são promovidas para repositórios de longa permanência (*common stores*). Neste estágio, o KMS atua como um *buffer*, isolando a rede de dados das flutuações da geração óptica [Mehic et al. 2020, ITU-T 2019]. Em topologias em que as aplicações não possuem conectividade óptica direta entre si, o material entra em uma fase de revezamento. Orientados pela camada de controle da QKDN, os nós intermediários utilizam as chaves armazenadas localmente para cifrar e retransmitir a chave de serviço salto a salto. Esse processo assegura que o trânsito da chave até o destino ocorra sem que o conteúdo da chave transite em claro pela rede [ITU-T 2019].

Ao atingir os nós de borda ou quando o enlace direto é suficiente, inicia-se a fase de suprimento. Quando uma aplicação na camada de serviço, como instâncias de redes privadas virtuais, necessita de chaves, ela emite uma requisição ao KMS por meio de *Application Programming Interfaces (APIs)*. A chave e seus metadados cruzam a fronteira de demarcação de segurança (*security demarcation boundary*) que isola a QKDN da rede do usuário [ITU-T 2019]. Na rede clássica, o material quântico pode ser consumido nativamente ou injetado como uma chave simétrica, a ser mesclada a chaves derivadas de protocolos tradicionais, como IPsec, MACsec ou TLS. Essa abordagem híbrida eleva a segurança das sessões convencionais contra a decifragem quântica [Lai et al. 2023].

A etapa final do ciclo de vida é o descarte da chave. Assim que a chave é consumida pelas aplicações criptográficas ou o seu período de validade expira, as diretrizes da QKDN determinam que os bits sejam apagados dos *buffers* de armazenamento dos gerenciadores envolvidos [ITU-T 2019]. Essa destruição lógica impede a reutilização de chaves, mitigando ataques de análise estatística e preservando o sigilo adiante. Desse modo, garante-se que os dados cifrados no presente não possam ser expostos na hipótese de o hardware dos nós ser fisicamente comprometido no futuro [ITU-T 2019].

Diante da visão geral do ciclo de vida de uma chave em QKDNs, torna-se evidente a necessidade de mecanismos padronizados que assegurem a interoperabilidade, a entrega segura e o gerenciamento eficiente do material criptográfico. Nesse contexto, o ETSI, por meio do *Industry Specification Group on Quantum Key Distribution (ISG QKD)*, desempenha um papel fundamental na definição de interfaces e protocolos que viabilizam a integração da QKD com aplicações e infraestruturas de comunicação clássicas. Destacam-se, em particular, as especificações ETSI GS QKD 004 e ETSI GS QKD 014, que estabelecem, respectivamente, a interface entre sistemas de gerenciamento de chaves e aplicações e uma API baseada em REST para a entrega de chaves criptográficas. Essas normas fornecem os elementos necessários para a implementação de soluções interoperáveis e seguras, essenciais para a operacionalização de serviços baseados em QKD. Assim, as subseções a seguir apresentam uma análise detalhada dessas especificações, enfatizando suas funcionalidades, modelos operacionais e contribuições para a consolidação de redes quânticas seguras [Sáez et al. 2024, James et al. 2023].

4.3. Interfaces Padronizadas de KMS: ETSI ISG QKD

Esta seção apresenta a abordagem elaborada pela *Industry Specification Group* (ISG) da *European Telecommunications Standards Institute* (ETSI), que posiciona o KMS como serviço acessível via API, promovendo a separação entre a infraestrutura QKD e as aplicações. Introduz as entidades fundamentais do modelo ETSI: KME e SAE, detalha as especificações ETSI GS QKD 004, que definem a interface entre KMS e a aplicação, e ETSI GS QKD 014, que padroniza uma API REST para a entrega de chaves em modelo cliente-servidor. A seção conclui ao comparar as abordagens ITU-T (gestão como função de rede) e ETSI (gestão como serviço), destacando sua complementaridade.

4.3.1. Abordagem ETSI: KMS como serviço

A arquitetura da ETSI para distribuição quântica de chaves define o KMS como uma camada de abstração essencial localizada entre os módulos QKD físicos e as diversas aplicações de usuário [ETSI 2020].

Essa arquitetura é governada por uma visão *API-centric*: toda interação entre aplicações e o sistema de distribuição de chaves deve ocorrer exclusivamente por meio de APIs, independentemente do fabricante do hardware quântico ou do protocolo óptico utilizado. Essa escolha de projeto garante interoperabilidade entre equipamentos de diferentes fornecedores e desacopla o desenvolvimento de aplicações da evolução da infraestrutura física. A ETSI materializou essa visão em duas normas complementares: a ETSI GS QKD 004, orientada a fluxos contínuos de chaves, e a ETSI GS QKD 014, baseada em REST para entregas sob demanda, ambas detalhadas nas seções seguintes [ETSI 2020, ETSI 2019].

O papel fundamental do KMS é gerenciar a sincronização, o armazenamento e a exclusão de chaves, entregando-as sob demanda, como um serviço, para aplicações pares nos pontos terminais da comunicação [ETSI 2019]. Para isso, o modelo ETSI organiza a rede em duas entidades fundamentais com responsabilidades bem delimitadas: a *Key Management Entity* e a *Secure Application Entities*.

- A KME é responsável por gerenciar, armazenar e entregar chaves simétricas geradas pelos módulos QKD físicos conectados a ela. Em termos arquiteturais, a KME atua como intermediária entre o hardware quântico que produz o material criptográfico bruto por meio de protocolos como o BB84[1] e as aplicações que consomem esse material por meio de APIs padronizadas [ETSI 2020]. Do ponto de vista operacional, cada KME mantém um reservatório de chaves (*key pool*) sincronizado com a KME do nó par, garantindo que ambos os lados de uma comunicação segura disponham do mesmo material criptográfico sem que ele precise ser transmitido pela rede clássica. Uma KME pode servir simultaneamente múltiplas SAEs, controlando o acesso e evitando o reuso indevido de chaves já entregues [ETSI 2019]. Formalmente, uma KME é identificada por um `KME_ID` único na rede, e cada par

[1]O protocolo BB84, proposto por Bennett e Brassard em 1984, é o protocolo de distribuição quântica de chaves mais amplamente estudado e implementado. Ele utiliza propriedades de polarização de fótons para estabelecer uma chave secreta compartilhada entre duas partes, com segurança garantida pelas leis da mecânica quântica.

de KMEs que compartilha um enlace quântico estabelece entre si uma associação de chaves que serve de base para todas as requisições das aplicações conectadas a elas [ETSI 2019].

- A SAE é uma aplicação ou sistema que consome chaves criptográficas providas pela KME local. Exemplos típicos de SAEs incluem roteadores IPsec, servidores TLS, sistemas de criptografia de disco ou qualquer outro componente que necessite de material criptográfico de alta entropia para proteger suas comunicações [ETSI 2020]. A SAE não possui conhecimento direto sobre o funcionamento do hardware quântico, os protocolos de distribuição de chaves ou a topologia da rede. Ela interage exclusivamente com sua KME local por meio da API padronizada, tratando o sistema QKD como uma *caixa-preta* que entrega chaves sob demanda. Essa separação de responsabilidades é uma das principais vantagens arquiteturais do modelo ETSI: aplicações podem ser desenvolvidas e testadas de forma independente da infraestrutura quântica subjacente [ETSI 2020].

A interação entre KME e SAE é estruturada por meio de associações vínculos lógicos que identificam unicamente uma sessão de entrega de chaves entre um par de SAEs. O identificador dessas associações varia conforme a norma utilizada: na ETSI GS QKD 004, é denominado *Key Stream ID* (KSID); na ETSI GS QKD 014, cada chave individual recebe um *Key ID*. Ambos são implementados no formato UUID v4[2], garantindo unicidade sem necessidade de coordenação centralizada [ETSI 2020, ETSI 2019].

Vale destacar uma premissa arquitetural importante: O modelo de confiança do ETSI assume que a comunicação entre uma SAE e sua KME local ocorre dentro de um perímetro de segurança controlado tipicamente a mesma instalação física ou rede local protegida. A norma não define mecanismos de segurança para esse enlace local, delegando essa responsabilidade à infraestrutura de rede do operador [ETSI 2020, ETSI 2019]. Essa premissa é importante: ela significa que a segurança fim-a-fim do sistema QKD depende não apenas das propriedades quânticas do canal óptico, mas também da integridade física e lógica dos nós que compõem a rede.

Com as entidades e o modelo de confiança estabelecidos, é possível detalhar a primeira interface padronizada que governa sua interação: a norma ETSI GS QKD 004, que define o ciclo de vida completo de uma associação de chaves por meio de chamadas de função estruturadas.

4.3.2. ETSI GS QKD 004

A norma ETSI GS QKD 004 define a interface por meio da qual as aplicações interagem com o KMS para solicitar e consumir material criptográfico [ETSI 2020]. Como ilustrado na Figura 4.1, o KMS coordena a entrega de chaves simétricas idênticas em ambos os pontos terminais de uma sessão segura, abstraindo os aplicativos e toda a complexidade da camada quântica subjacente. Para isso, a norma estrutura essa interação em torno de três operações que definem o ciclo de vida completo de uma associação de chaves.

[2]UUID (*Universally Unique Identifier*) versão 4 é um identificador de 128 bits gerado aleatoriamente, padronizado pela RFC 4122. Sua probabilidade de colisão é suficientemente baixa para uso prático em sistemas distribuídos.

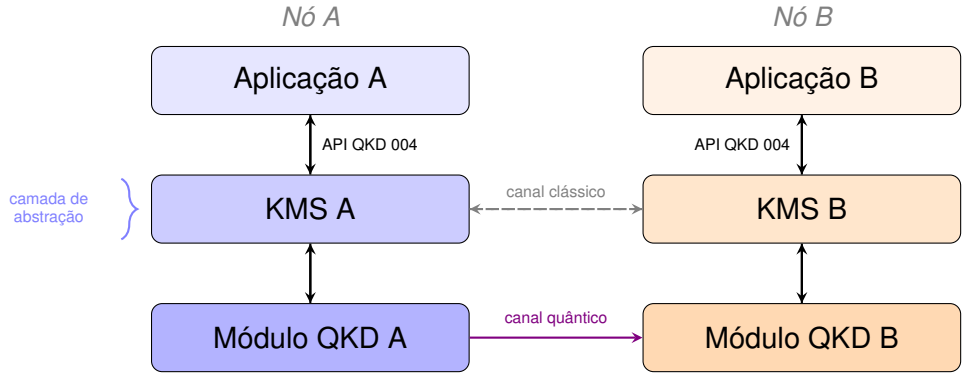


Figura 4.1. Arquitetura em camadas da norma ETSI GS QKD 004: o KMS atua como camada de abstração entre os módulos QKD físicos e as aplicações finais.

O ciclo se inicia com `OPEN_CONNECT`, que reserva uma associação identificada pelo KSID introduzido na seção anterior, um identificador que referencia o fluxo de bits sincronizados entre os pares, mas não contém nem permite derivar material criptográfico em si. Uma vez aberta a associação, `GET_KEY` permite que a aplicação recupere o material de chave correspondente: a sincronização entre os pares é garantida por um parâmetro de índice (*index*), que especifica a posição exata da chave a ser extraída no reservatório reservado, assegurando que ambos os lados obtenham o mesmo material. Por fim, `CLOSE` encerra a associação e descarta as chaves não utilizadas, seja ao término da sessão ou por expiração do tempo de vida definido, prevenindo o acúmulo de material criptográfico ocioso. A Figura 4.2 ilustra esse fluxo completo.

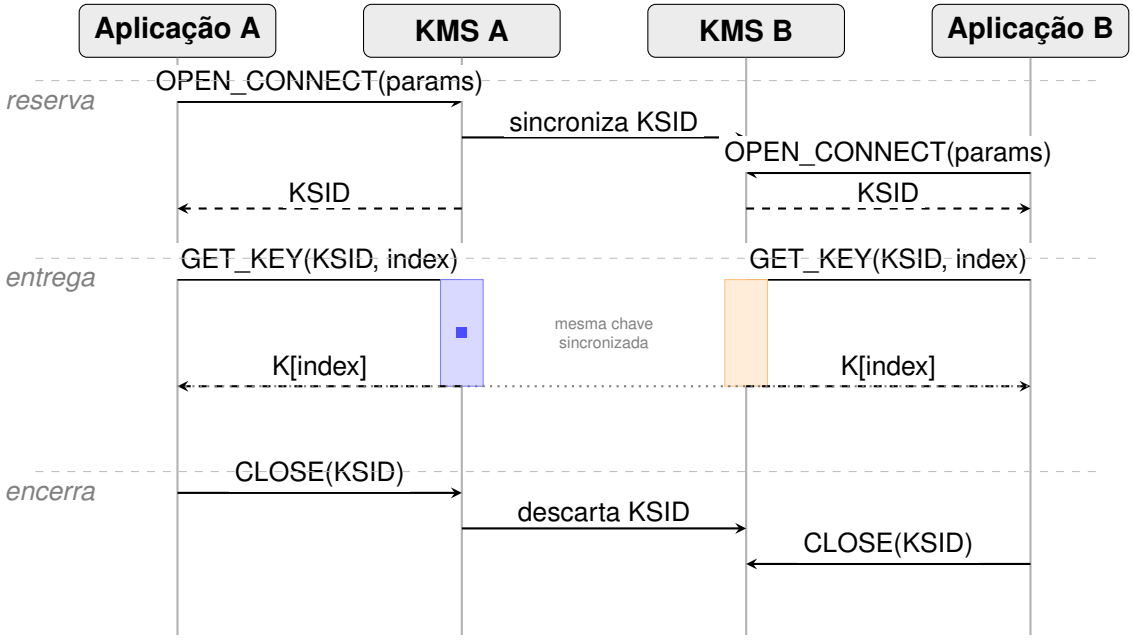


Figura 4.2. Diagrama de sequência da norma ETSI GS QKD 004: ciclo de vida completo de uma associação de chaves, desde a reserva via `OPEN_CONNECT` até o encerramento via `CLOSE`.

Aplicações distintas têm requisitos distintos de taxa, latência e janela de exposi-

ção criptográfica: um sistema de criptografia de enlaces de *backbone* exige altíssima taxa de bits e baixíssima variação temporal, enquanto uma aplicação de autenticação pontual pode tolerar taxas menores, desde que as chaves expirem rapidamente após o uso. Para acomodar essa diversidade, a norma prevê parâmetros de *Quality of Service (QoS)* negociados durante a chamada `OPEN_CONNECT`, organizados em dois grupos. O primeiro governa o desempenho: o tamanho do bloco de chave (*Key_chunk_size*), os limites de taxa de bits (*Max_bps* e *Min_bps*), a variação temporal na entrega (*Jitter*) e o nível de prioridade da requisição frente a outras em curso. O segundo governa o ciclo de vida do material criptográfico: o parâmetro Time to Live (TTL) determina por quanto tempo as chaves podem permanecer armazenadas na aplicação antes de serem descartadas, limitando a janela de exposição; os metadados complementares, como a idade da chave e o número de saltos percorridos na rede, fornecem dados de diagnóstico e estatísticas operacionais sobre a infraestrutura QKD subjacente [ETSI 2020].

A norma oferece, portanto, controle granular sobre todo o ciclo de vida das chaves e sobre os parâmetros de qualidade de serviço, mas pressupõe familiaridade com interfaces de baixo nível orientadas a chamadas de função. Para cenários que demandam uma integração mais ágil e acessível aos desenvolvedores de aplicações web, a ETSI propôs uma segunda interface, apresentada a seguir.

4.3.3. ETSI GS QKD 014: API REST para Entrega de Chaves

Diferente da abordagem orientada a fluxos contínuos da norma anterior, esta norma, ETSI GS QKD 014, define um protocolo de comunicação e formato de dados voltados à entrega de chaves por meio de uma interface REST, aproveitando convenções já consolidadas no desenvolvimento web [ETSI 2019]. Todas as interações utilizam HTTPS (TLS 1.2 ou superior), garantindo confidencialidade e integridade no transporte, e as requisições e respostas são codificadas em JSON, formato leve e amplamente suportado pelas principais linguagens e bibliotecas de programação. A API expõe três operações centrais: `Get status`, para verificar a disponibilidade de chaves no KME; `Get key`, para solicitar novas chaves; e `Get key with key IDs`, para recuperar chaves específicas a partir de seus identificadores.

O modelo de operação estrutura-se como um ciclo de requisição e resposta entre SAE e KME, com dois papéis complementares. A **Master SAE** é a aplicação que inicia a requisição de novas chaves à sua KME local e, além do material criptográfico, recebe os *Key IDs* associados a cada chave. A **Slave SAE**, após ser notificada do *Key ID* pela *Master SAE* por meio de um canal externo, utiliza esse identificador para solicitar a mesma chave à sua KME local, garantindo que ambos os lados compartilhem o mesmo material criptográfico sem que ele trafegue diretamente pela rede clássica. A Figura 4.3 ilustra o fluxo completo dessa troca, desde a solicitação inicial até a recuperação simétrica da chave.

Por adotar ferramentas familiares ao ecossistema web, a norma 014 apresenta uma barreira de entrada significativamente menor do que a 004. Enquanto a norma 004 pode ser implementada em C sem quaisquer dependências externas, a 014 se beneficia de *frameworks* de alto nível, acelerando a criação de protótipos e provas de conceito. Isso a torna especialmente adequada para ambientes de nuvem e aplicações que não exigem

o gerenciamento contínuo de fluxos de chaves, cenários em que entregas pontuais sob demanda por chamadas HTTP são suficientes. Seu objetivo central é, portanto, viabilizar a interoperabilidade entre equipamentos de diferentes fabricantes de forma simples e escalável, ampliando o ecossistema de adoção do QKD para além dos ambientes de telecomunicações tradicionais [ETSI 2019].

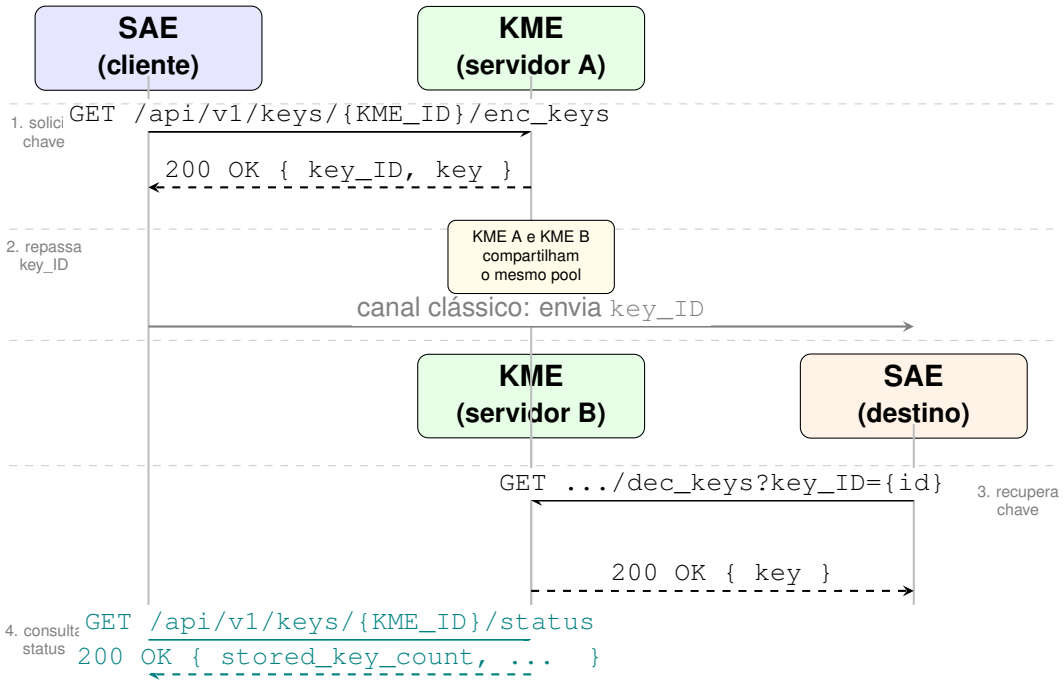


Figura 4.3. Fluxo de comunicação REST da norma ETSI GS QKD 014: a SAE cliente obtém a chave via `enc_keys`, repassa o `key_ID` pelo canal clássico e a SAE destino recupera a mesma chave via `dec_keys`.

A evolução da padronização da QKD evidencia uma abordagem complementar entre organismos internacionais distintos. Estudos recentes, como [Sáez et al. 2024], indicam que a ETSI tem desempenhado um papel fundamental na definição de interfaces, modelos de interoperabilidade e mecanismos de entrega de chaves para aplicações, promovendo a integração da QKD com infraestruturas criptográficas convencionais, como evidenciado nas especificações do ETSI ISG QKD e em análises sobre o estado da padronização da tecnologia. Em contrapartida, a ITU-T concentra-se na especificação de arquiteturas de rede, planos de controle e estruturas de gerenciamento capazes de sustentar a operação de redes QKDN em larga escala. Trabalhos como [James et al. 2023] sobre sistemas de gerenciamento de chaves em redes QKD reforçam essa distinção ao destacar a necessidade de arquiteturas escaláveis e alinhadas aos modelos da série ITU-T Y.3800 [ITU-T 2019]. Nesse contexto, a consolidação de um ecossistema global seguro depende da convergência entre essas iniciativas normativas, especialmente quanto à escalabilidade, à interoperabilidade e à gestão do material criptográfico. Assim, a próxima seção apresenta uma visão detalhada das recomendações da série ITU-T Y.3800, destacando seus modelos arquiteturais e mecanismos de controle e de gerenciamento para redes de distribuição quântica de chaves.

4.4. KMS em Redes QKD: Padronização pela Série ITU-T Y.3800

A evolução da QKD de enlaces ponto a ponto para infraestruturas de rede desloca o problema da segurança quântica do nível do enlace para o nível da arquitetura, da operação e da interoperabilidade [Sanz et al. 2025]. Quando a geração quântica de chaves precisa atender múltiplos nós, suportar retransmissão segura, integrar-se a aplicações criptográficas convencionais e operar de forma coordenada em ambientes heterogêneos, torna-se necessário definir modelos padronizados para funções, interfaces e responsabilidades ao longo da rede. Nesse contexto, a série ITU-T Y.3800 estabelece a base normativa para QKDNs, fornecendo uma visão geral da rede, requisitos funcionais, arquitetura funcional, mecanismos de gerenciamento de chaves e funções de controle e gerenciamento [ITU-T 2019, ITU-T 2020a, ITU-T 2020d, ITU-T 2020b].

A série ITU-T Y.3800 organiza a padronização de redes de distribuição quântica de chaves como um conjunto progressivo de recomendações voltadas à inserção da QKD em infraestruturas de telecomunicações. No âmbito da ITU-T, esse trabalho está associado ao *Study Group* 13 (SG13), responsável por estudos sobre arquiteturas de redes futuras, no qual a QKDN é tratada como um problema de rede que abrange estrutura conceitual, requisitos funcionais, arquitetura, gerenciamento de chaves e mecanismos de controle e gerenciamento [ITU-T 2019]. Embora outros grupos, como o SG11 e o SG17, também dialoguem com temas relevantes, tais como protocolos, sinalização, segurança e o panorama evolutivo da padronização, os documentos por eles produzidos excedem o escopo deste módulo. Nesse contexto, a série Y.3800, desenvolvida no ecossistema normativo do SG13, estabelece a base arquitetural principal para QKDN.

Assim, sua organização pode ser compreendida de forma hierárquica, como ilustrado na Figura 4.4. A recomendação ITU-T Y.3800[3] apresenta a visão geral das redes que suportam QKD e delimita aspectos como a estrutura conceitual, o modelo em camadas e as funções básicas. A Y.3801 especifica os requisitos funcionais da QKDN, a Y.3802 formaliza sua arquitetura funcional, a Y.3803 detalha o gerenciamento de chaves, considerado central para a operação significativa da rede, e a Y.3804 define as funções e os procedimentos de controle e gerenciamento da QKDN [ITU-T 2019, ITU-T 2020a, ITU-T 2020c, ITU-T 2020d, ITU-T 2020b]. Para fins de delimitação temática, este módulo concentra-se na base conceitual, funcional e operacional estabelecida por essas recomendações fundamentais.

4.4.1. Motivação para padronização de redes QKD

Apesar das garantias teóricas de segurança da QKD, sua implementação prática permanece condicionada por restrições físicas e imperfeições dos dispositivos. Em sistemas reais, perdas em canais ópticos, ruído de detecção, limitações de fontes e vulnerabilidades de implementação afetam a taxa de geração de chaves e restringem o alcance útil dos enlaces diretos. Como consequência, a operação de QKD em ampla escala não pode ser tratada apenas como um problema de prova de segurança do protocolo, mas também como um problema de arquitetura e operação de rede [Xu et al. 2020].

Essas limitações tornam insuficiente o modelo baseado exclusivamente em enlaces

[3]Doravante chamada apenas de Y.38XX.

Figura 4.4. Estrutura da série ITU-T Y.3800 para QKDNs, evidenciando a relação entre a visão geral (Y.3800), os requisitos funcionais (Y.3801), a arquitetura funcional (Y.3802), o gerenciamento de chaves (Y.3803) e os mecanismos de controle e gerenciamento da rede (Y.3804).

ponto a ponto [Narroway 2025, Ahmed et al. 2025]. Em enlaces isolados, a geração de chaves ocorre apenas entre nós diretamente conectados, o que restringe a conectividade, dificulta o atendimento a múltiplos usuários e reduz a flexibilidade operacional da infraestrutura. Além disso, a ausência prática de repetidores quânticos escaláveis faz com que, no estado atual da tecnologia, a expansão do alcance dependa de arquiteturas multi-nó com retransmissão por nós confiáveis, o que desloca o foco da simples geração de chaves para problemas de armazenamento, sincronização, retransmissão, roteamento e provisão de serviço [Cao et al. 2022, Mehic et al. 2020].

Nesse cenário, a evolução para redes QKD não responde apenas à necessidade de ampliar cobertura, mas também à necessidade de prover escalabilidade, robustez e continuidade operacional. Uma QKDN pode explorar caminhos alternativos para distribuição de chaves, suportar múltiplas aplicações e integrar diferentes tecnologias e equipamentos sob uma camada lógica comum de gerenciamento. Isso torna a padronização um requisito estrutural para garantir interoperabilidade, operação consistente e integração com redes de usuário. É nesse contexto que a série ITU-T Y.3800 se estabelece como framework normativo para QKDN, definindo estrutura conceitual, requisitos funcionais, arquitetura, gerenciamento de chaves e mecanismos de controle e gerenciamento da rede [ITU-T 2019, ITU-T 2020a, ITU-T 2020c, ITU-T 2020d, ITU-T 2020b].

4.4.2. Visão arquitetural das QKDN

A recomendação ITU-T Y.3800 estabelece a visão geral de redes que suportam QKD e delimita os aspectos necessários para sua implementação. Seu escopo abrange uma visão das tecnologias QKD, as capacidades de rede para seu suporte, a estrutura conceitual da QKDN e suas funções básicas [ITU-T 2019]. Com isso, a recomendação desloca a discussão da QKD do nível estrito do enlace físico para o nível da arquitetura de rede, no qual a geração quântica de chaves precisa ser articulada com funções de gerenciamento, controle e provisão de serviço.

A Y.3800 assume que a QKD é uma tecnologia adicional às redes de comunicação, e não um substituto completo da infraestrutura de telecomunicações. Em outras palavras, a QKDN não elimina a rede clássica, mas a complementa ao prover material criptográfico simétrico com garantias associadas à mecânica quântica. Essa formulação é importante porque situa a QKD no interior de uma arquitetura híbrida, na qual enlaces quânticos,

enlaces clássicos, funções de rede e aplicações criptográficas precisam operar de forma coordenada [ITU-T 2019].

Um modelo de comunicação elementar baseado em QKD conta com enlace ponto a ponto entre dois módulos QKD, como ilustrado na Figura 4.5. Esses módulos estão conectados por um canal quântico e por um canal clássico autenticado. Nesse arranjo, as chaves geradas são entregues localmente às aplicações criptográficas em cada extremidade. Embora esse modelo seja adequado para demonstrar o funcionamento da tecnologia, ele não atende às demandas de ambientes com múltiplos nós, usuários e requisitos de escalabilidade. É precisamente para superar essa limitação que a Y.3800 introduz o conceito de QKDN, definido como uma rede composta por dois ou mais nós QKD conectados por enlaces QKD, permitindo o compartilhamento de chaves entre nós arbitrários por meio de funções adicionais de gerenciamento e retransmissão [ITU-T 2019].

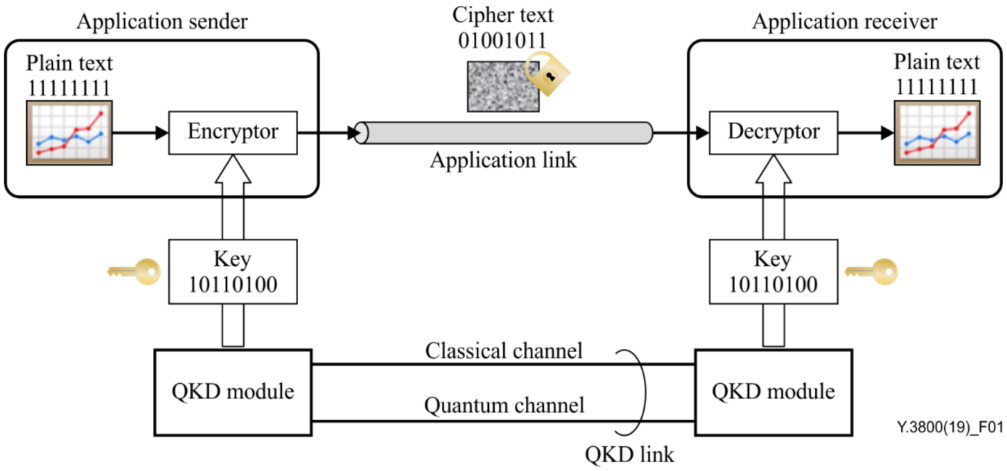


Figura 4.5. Exemplo de uso de QKD para proteção de um enlace de aplicação ponto a ponto.

Fonte: Figura 1 ITU-T Y.3800 [ITU-T 2019].

Um dos pontos centrais da Y.3800 é a distinção entre a QKDN e rede de usuário (*user network* na série). A QKDN tem por função produzir, armazenar, retransmitir e fornecer chaves. A rede do usuário, por sua vez, consome essas chaves em aplicações criptográficas, como cifragem, autenticação ou outros mecanismos de proteção. Essa separação é conceitualmente importante porque delimita duas responsabilidades distintas: a responsabilidade da infraestrutura QKDN sobre a gestão do material criptográfico e a responsabilidade da aplicação sobre o uso efetivo da chave recebida [ITU-T 2019].

A recomendação também introduz a noção de *security demarcation boundary*, isto é, uma fronteira de responsabilidade entre a QKDN e a rede do usuário. Essa fronteira é relevante porque ajuda a organizar interfaces de fornecimento de chaves, requisitos de autenticação e mecanismos de gerenciamento. Na prática, ela permite que desenvolvedores de aplicações tratem a chave como um recurso recebido da rede, sem necessidade de conhecer os detalhes internos do processo de geração e *relay* dentro da QKDN, ao mesmo tempo em que preserva a autonomia da infraestrutura QKDN quanto às suas políticas internas de armazenamento, ciclo de vida e retransmissão [ITU-T 2019].

A Y.3800 reconhece que a QKD é, por natureza, uma tecnologia ponto a ponto

sujeita a restrições físicas de alcance. Para ampliar a conectividade e a disponibilidade, a recomendação discute diferentes mecanismos de formação de rede, incluindo comutação ou divisão óptica, retransmissão por nós confiáveis (*trusted relaying*), retransmissão assistida por medição e, em uma perspectiva de longo prazo, redes totalmente quânticas com repetidores quânticos [ITU-T 2019]. No entanto, sob a perspectiva da implementação prática, a recomendação concentra sua análise em QKDNs baseadas em nós confiáveis.

Assim, os nós intermediários armazenam e retransmitem chaves entre nós adjacentes da rede, viabilizando o mecanismo de *key relay*, pelo qual chaves podem ser compartilhadas entre nós sem enlace QKD direto. Essa solução amplia o alcance e a flexibilidade topológica da rede, mas faz com que os nós intermediários passem a integrar seu domínio de confiança. Desse modo, a segurança do sistema deixa de depender apenas da integridade do enlace quântico e passa a exigir também a proteção física e lógica desses nós, o gerenciamento do ciclo de vida das chaves e a coordenação entre as funções de *relay* e *supply* [ITU-T 2019].

Uma das principais contribuições da recomendação ITU-T Y.3800 para a arquitetura de QKDN é a definição de um modelo em camadas e de um conjunto de funções básicas. A recomendação organiza as capacidades de rede necessárias ao suporte da QKD em uma estrutura conceitual composta pela camada quântica, pela camada de gerenciamento de chaves, pela camada de controle da QKDN e pela camada de gerenciamento da QKDN, além de sua relação com a camada de serviço e com a rede do usuário [ITU-T 2019].

Ainda seguindo a recomendação, a QKDN, no nível das funções básicas, deve distribuir suas principais capacidades entre as diferentes camadas. À camada quântica cabem a geração de chaves e a operação dos módulos QKD e seus enlaces. À camada de gerenciamento de chaves são atribuídas funções como redimensionamento, reformatação com metadados, armazenamento, *relay*, sincronização, autenticação e fornecimento de chaves às aplicações criptográficas. A camada de controle, por sua vez, compreende o controle de rotas de *relay*, a reconfiguração diante de falhas ou tentativas de interceptação, o controle de módulos e enlaces, além de funções de autenticação, autorização e aplicação de políticas de QoS. Por fim, a camada de gerenciamento abrange funções de monitoramento global, suporte a FCAPS (*Fault, Configuration, Accounting, Performance and Security*), gerenciamento do ciclo de vida das chaves e suporte aos mecanismos de autenticação e autorização [ITU-T 2019]. Essa decomposição é fundamental por antecipar, em nível conceitual, aspectos que serão posteriormente refinados nas recomendações subsequentes.

Em síntese, a Y.3800 fornece a moldura arquitetural. Define-se o problema de rede que a QKDN pretende resolver, delimita a relação entre infraestrutura quântica e aplicações, explicita a necessidade de *trusted nodes* e de *key relay* e estabelece a estrutura em camadas sobre a qual se apoiam os requisitos funcionais da Y.3801 e a arquitetura funcional formalizada em Y.3802.

4.4.3. Requisitos funcionais da QKDN

A recomendação ITU-T Y.3801 tem por objetivo especificar os requisitos funcionais de uma QKDN. Seu escopo abrange quatro camadas de requisitos (Figura 4.6), correspondentes à camada quântica, à camada de gerenciamento de chaves, à camada de controle

da QKDN e à camada de gerenciamento da QKDN [ITU-T 2020a]. Enquanto a Y.3800 define a estrutura conceitual e as funções básicas da arquitetura, a Y.3801 detalha os requisitos mínimos que devem ser atendidos para assegurar o funcionamento coerente da rede.

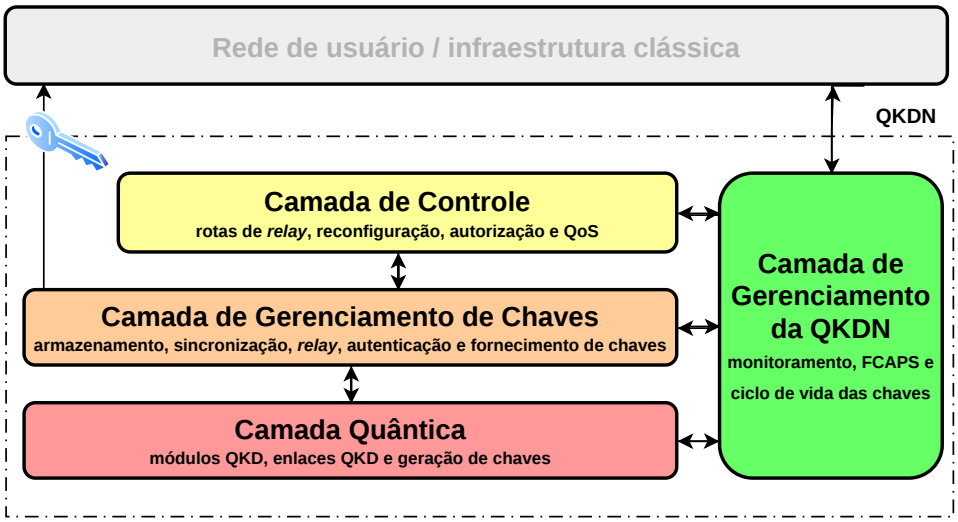


Figura 4.6. Arquitetura funcional em camadas de uma QKDN, com base na ITU-T Y.3802.

Requisitos da camada quântica: Na camada quântica, a Y.3801 estabelece que os protocolos QKD adotados devem ser comprovadamente seguros no modelo considerado e aptos a viabilizar o estabelecimento de chaves com segurança teórica da informação. Para isso, o módulo QKD deve implementar as funções necessárias à execução de um ou mais protocolos QKD com o módulo correspondente conectado por um enlace QKD, incluindo, entre outras, geração de números aleatórios, comunicação quântica, destilação de chaves e sincronização do canal [ITU-T 2020a]. A recomendação também determina que esse módulo esteja contido em um limite criptográfico bem definido, uma vez que a segurança da camada quântica depende não apenas do protocolo em abstrato, mas também da delimitação do domínio em que seus componentes e processos relevantes permanecem protegidos. Além disso, os pares de módulos QKD devem transferir as chaves geradas aos respectivos *Key Managers* (KM) por meio de interfaces apropriadas, bem como disponibilizar informações de estado, falha e desempenho às camadas superiores de controle e gerenciamento [ITU-T 2020a]. Assim, a camada quântica não se configura como um subsistema isolado, mas como um componente integrado à arquitetura da QKDN, cuja operação deve fornecer visibilidade suficiente para que a rede reaja adequadamente a degradações, falhas e eventos de segurança.

Requisitos da camada de gerenciamento de chaves: Na camada de gerenciamento de chaves, a Y.3801 estabelece os requisitos que permitem transformar as chaves geradas na camada quântica em recursos efetivamente utilizáveis por aplicações criptográficas. Nesse contexto, o *Key Manager* (KM) deve ser capaz de receber chaves dos módulos

QKD, armazená-las de forma segura quando necessário e formatá-las conforme as exigências de uso interno, de *relay* ou de *supply* [ITU-T 2020a]. A recomendação também prevê, de forma preferencial, compatibilidade com diferentes tipos de módulos QKD, favorecendo a interoperabilidade entre implementações heterogêneas. Além disso, o Key Manager (KM) deve disponibilizar informações de gerenciamento às funções de controle e gerenciamento da QKDN, bem como dados de falha e desempenho relativos a si próprio e aos enlaces de gerenciamento. Também se recomenda que suporte funções como *key relay*, *key supply* e gerenciamento do ciclo de vida das chaves [ITU-T 2020a]. Em conjunto, esses requisitos evidenciam que o KM não se limita a atuar como um repositório local, mas desempenha um papel central na articulação entre geração, armazenamento, rastreabilidade, sincronização e provisão do material criptográfico ao longo da rede.

Requisitos da camada de controle: Na Y.3801, a camada de controle deve coordenar os recursos da rede de modo a assegurar uma operação segura, estável, eficiente e robusta. Entre seus principais requisitos, destacam-se a capacidade de provisionar e controlar rotas de *relay* entre os extremos que demandam chaves, rerotear essas rotas de acordo com o estado da camada quântica e da camada de gerenciamento de chaves, controlar a configuração de módulos e enlaces e aplicar, quando pertinente, políticas de QoS e de cobrança [ITU-T 2020a]. Exigem-se mecanismos de autenticação e autorização para os elementos da rede, aspecto particularmente relevante em QKDNs baseadas em nós confiáveis, nas quais o domínio de confiança da arquitetura depende do controle consistente de identidades, papéis e permissões dos componentes sob gerenciamento do sistema.

Requisitos da camada de gerenciamento: Na camada de gerenciamento, a Y.3801 define requisitos voltados ao suporte das funções clássicas de operação de rede, com ênfase em falhas, configuração, contabilidade, desempenho e segurança. Nesse contexto, a recomendação estabelece que o gerenciador da QKDN deve ser capaz de realizar o monitoramento de falhas, a coleta e a análise de dados de desempenho, a provisão e a configuração de recursos, bem como o gerenciamento de informações de segurança relevantes para a operação da rede [ITU-T 2020a]. Além disso, atribui a essa camada o suporte ao gerenciamento do ciclo de vida das chaves e à coordenação com a camada de controle. Em conjunto, esses requisitos reforçam que a QKDN é concebida, no âmbito da série Y.3800, como uma infraestrutura de telecomunicações em sentido pleno, na qual as chaves constituem o recurso central, mas sua utilidade depende diretamente de mecanismos explícitos de observabilidade, controle, manutenção e governança operacional.

Uma das principais contribuições da Y.3801 é transformar a decomposição conceitual da Y.3800 em um conjunto de exigências funcionais que orientam a implementação de rede. A recomendação não define ainda a arquitetura funcional detalhada de cada entidade, mas prepara o terreno para isso ao explicitar o que cada camada precisa fazer para que a QKDN seja operacionalmente viável. Desse modo, a Y.3801 funciona como elo normativo entre a visão arquitetural geral da Y.3800 e a formalização da arquitetura funcional apresentada em Y.3802.

4.4.4. Arquitetura funcional da QKDN

A recomendação ITU-T Y.3802 formaliza a arquitetura funcional das redes de distribuição de chaves quânticas. Seu escopo abrange o modelo funcional da QKDN, os elementos funcionais de cada camada, os pontos de referência entre entidades, as configurações arquiteturais possíveis e os procedimentos operacionais básicos da rede [ITU-T 2020c]. Em relação à Y.3800, que define a visão geral e a estrutura conceitual da QKDN, e à Y.3801, que explicita os requisitos funcionais por camada, a Y.3802 representa a etapa em que a rede passa a ser descrita em termos de entidades funcionais, interfaces e modos de organização arquitetural.

Do ponto de vista da engenharia de redes, a importância da Y.3802 reside em transformar a QKDN em um modelo sistemático de composição funcional. Em vez de tratar a QKD apenas como um conjunto de enlaces quânticos e dispositivos especializados, a recomendação organiza a rede em camadas articuladas, cada uma com funções próprias e interfaces explícitas. Com isso, a distribuição quântica de chaves deixa de ser vista apenas como processo físico de geração de material criptográfico e passa a ser tratada como um serviço de rede sustentado por mecanismos de gerenciamento, controle e integração com aplicações [ITU-T 2020c].

A Y.3802 estabelece um modelo funcional organizado em seis camadas: camada quântica, camada de gerenciamento de chaves, camada de controle da QKDN, camada de gerenciamento da QKDN, camada de serviço e camada de gerenciamento da rede do usuário [ITU-T 2020c]. Essa organização preserva a coerência com a estrutura conceitual proposta na Y.3800, mas a aprofunda ao explicitar entidades funcionais, subfunções internas e pontos de referência para a troca de informações entre os diferentes componentes da arquitetura. Nesse modelo, a camada quântica reúne os módulos QKD e os enlaces QKD responsáveis pela geração de chaves; a camada de gerenciamento de chaves encarrega-se de recebê-las, tratá-las, armazená-las, retransmiti-las e fornecê-las às aplicações; a camada de controle coordena os recursos da rede, as rotas de *relay* e os mecanismos de reconfiguração operacional; e a camada de gerenciamento provê supervisão global, coleta de informações operacionais e suporte às funções de controle. Complementarmente, a camada de serviço abriga as aplicações criptográficas que consomem as chaves disponibilizadas pela QKDN, enquanto a camada de gerenciamento da rede do usuário representa a interface com o ambiente administrativo e operacional da rede na qual essas aplicações se inserem [ITU-T 2020c].

A Figura 4.6 evidencia que a arquitetura funcional da QKDN vai além de uma cadeia linear entre a geração e o consumo de chaves. Em vez disso, ela se organiza como uma infraestrutura em que o material criptográfico circula entre funções especializadas, acompanhado por informações de estado e desempenho, e cuja operação depende de mecanismos de controle e gerenciamento capazes de assegurar disponibilidade, rastreabilidade e coerência operacional. A separação em camadas é particularmente relevante nesse contexto, pois permite integrar a QKDN a redes clássicas preservando a distinção entre, de um lado, os processos físicos associados à QKD e, de outro, as funções de rede voltadas ao *relay*, ao *supply*, à supervisão e à coordenação do sistema.

Elementos funcionais da camada quântica: Na camada quântica, a Y.3802 descreve o módulo QKD como uma entidade composta por um conjunto de funções responsáveis tanto pela execução do protocolo quanto pela operação do enlace quântico. Entre elas, incluem-se a função de comunicação quântica, encarregada de preparar, transmitir e medir sinais quânticos; a função de sincronização do canal quântico; a função de destilação de chaves, que compreende etapas como *sifting*, estimação de parâmetros, correção de erros e amplificação de privacidade; a função de fornecimento de chaves QKD; a função de geração de números aleatórios; e a função de controle e gerenciamento do próprio módulo [ITU-T 2020c].

Essa decomposição é relevante porque explicita que a geração de chaves não é uma operação monolítica. No interior da camada quântica, a obtenção de material criptográfico seguro depende de subfunções com papéis distintos, cada uma sujeita a requisitos próprios de sincronização, processamento e monitoramento. Além disso, a recomendação admite funções opcionais, como multiplexação de canais, comutação ou divisão óptica e pontos de *relay* quântico, o que mostra que a camada quântica pode assumir diferentes formas de implementação sem abandonar a arquitetura funcional geral [ITU-T 2020c].

Elementos funcionais da camada de gerenciamento de chaves: A camada de gerenciamento de chaves ocupa posição central na Y.3802. Assim, o Key Manager deixa de ser tratado como caixa lógica única e passa a ser decomposto em duas entidades funcionais: a *Key Management Agent (KMA)* e a *Key Supply Agent (KSA)* [ITU-T 2020c]. Essa distinção tem impacto arquitetural, pois organiza de forma clara a trajetória da chave entre a camada quântica e a aplicação consumidora.

A KMA é responsável por adquirir chaves dos módulos QKD, sincronizá-las, redimensioná-las, reformatá-las com metadados, armazená-las e retransmiti-las por enlaces apropriados. Nessa entidade se concentram, portanto, as funções associadas à gestão interna do material criptográfico na QKDN. A KSA, por sua vez, atua entre a KMA e a aplicação criptográfica. Sua função é sincronizar e autenticar as chaves compartilhadas entre as extremidades e fornecê-las sob demanda às aplicações por meio da interface apropriada [ITU-T 2020c]. A recomendação também prevê, de forma opcional, a função de combinação de chaves, relevante em cenários híbridos em que a QKD pode ser associada a outros métodos de troca ou derivação de chaves.

Essa decomposição confirma que o gerenciamento de chaves é o elo estrutural entre produção quântica e provisão de serviço. Em termos de arquitetura, a QKDN só se torna operacionalmente útil porque existe uma camada dedicada a transformar sequências simétricas produzidas por módulos QKD em chaves utilizáveis por aplicações distribuídas, sob políticas de armazenamento, identificação, sincronização, *relay* e *supply*.

Elementos funcionais das camadas de controle e de gerenciamento: Na camada de controle, a Y.3802 especifica funções como controle de sessão, controle de roteamento, controle de configuração, controle baseado em políticas, controle de acesso e controle e gerenciamento do próprio controlador [ITU-T 2020c]. Em conjunto, essas funções permitem coordenar o estabelecimento dos fluxos de chaves na rede, definir rotas de *relay*

entre nós, reagir a falhas e administrar recursos de modo compatível com requisitos de QoS e de segurança.

Na camada de gerenciamento, a recomendação organiza as funções segundo o modelo FCAPS, contemplando o gerenciamento de falhas, a configuração, a contabilidade, o desempenho e a segurança [ITU-T 2020c]. Além disso, essa camada desempenha um papel transversal de suporte à operação da QKDN, incluindo o acompanhamento de seu estado global, o registro de informações de desempenho, o apoio às decisões da camada de controle e o suporte ao gerenciamento do ciclo de vida das chaves. Em conjunto, as camadas de controle e gerenciamento conferem à QKDN características de infraestrutura de telecomunicações, e não apenas de um arranjo local de dispositivos quânticos.

Pontos de referência: Na Y.3802, a interação entre as diferentes entidades funcionais da QKDN é descrita por meio de pontos de referência. Em termos arquiteturais, um ponto de referência corresponde a uma interface funcional que delimita onde e como ocorre a troca de informações entre dois componentes do sistema. Com base nesse conceito, a recomendação especifica pontos de referência nos módulos QKD, no KM, no controlador da QKDN, no gerenciador da QKDN, no gerenciamento da rede do usuário e nas aplicações criptográficas [ITU-T 2020c]. Esses pontos de referência representam, portanto, interfaces padronizadas para a troca de informações de sincronização, gerenciamento, controle, autenticação e fornecimento de chaves.

Para o propósito deste capítulo, o ponto mais importante não é a listagem exaustiva de todos os identificadores, mas a compreensão de seu papel arquitetural. Ao explicitar interfaces funcionais, a Y.3802 estabelece as condições para interoperabilidade entre componentes, separação de responsabilidades entre camadas e integração entre a infraestrutura QKDN e a rede do usuário. Em particular, a interface entre a KSA e a aplicação criptográfica é a materialização arquitetural do momento em que a chave deixa de estar sob responsabilidade exclusiva do gerenciamento interno da QKDN e passa a ser consumida como recurso de serviço.

4.4.5. Configurações arquiteturais da QKDN

Além do modelo funcional geral, a Y.3802 descreve quatro configurações arquiteturais possíveis para a QKDN. Essas configurações evidenciam que a arquitetura funcional pode ser concretizada de diferentes formas, com distintos níveis de centralização e formas de distribuição das funções entre os nós da rede [ITU-T 2020c]. Embora tais variações não modifiquem a lógica fundamental da arquitetura, elas influenciam diretamente a maneira como as funções de controle, *relay* e coordenação são organizadas e executadas no ambiente de rede.

Configuração 1 – QKDN distribuída: Nesta configuração, conforme apresentada da Figura 4.7, a QKDN adota uma organização distribuída, na qual cada nó do tipo 1 incorpora módulos QKD, um KM e um controlador da QKDN. Nesse arranjo, as funções de controle são executadas localmente, sem a necessidade de um controlador centralizado [ITU-T 2020c]. Essa configuração tende a favorecer maior autonomia operacional dos nós e maior resiliência à indisponibilidade de uma entidade central, mas também am-

plia a complexidade funcional de cada elemento da rede. Sob a perspectiva arquitetural, trata-se de uma alternativa adequada para cenários em que se busca reduzir dependências centralizadas e em que os nós dispõem de capacidade suficiente para executar localmente funções de controle, roteamento de *relay* e reconfiguração. Em contrapartida, em redes mais extensas ou com topologias mais dinâmicas, essa abordagem pode tornar mais complexa a coordenação global das decisões operacionais.

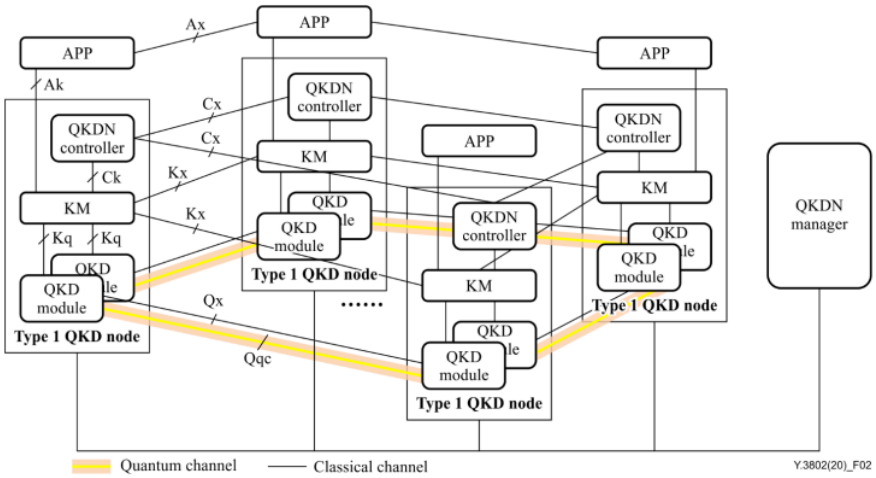


Figura 4.7. Configuração 1 da Y.3802, correspondente a uma QKDN distribuída, na qual cada nó integra módulos QKD, gerenciamento de chaves e funções de controle.

Fonte: Figura 2 ITU-T Y.3802 [ITU-T 2020c].

Configuração 2 – QKDN centralizada: já nesta configuração descrita pela Y.3802, as funções de controle são concentradas em um ou mais controladores da QKDN. A Figura 4.8 apresenta a configuração. Nesse arranjo, os nós do tipo 2 passam a abrigar apenas os módulos QKD e o KM, enquanto o controlador é tratado como uma entidade funcional separada na arquitetura [ITU-T 2020c]. Essa organização busca tornar o controle da rede mais eficiente ao deslocar decisões de roteamento, reconfiguração e aplicação de políticas para uma entidade com visão mais ampla da topologia e do estado global da QKDN. Como resultado, os nós tendem a se tornar funcionalmente mais simples, e a coordenação de *relay* e o uso global dos recursos da rede podem ser conduzidos de forma mais consistente, especialmente em cenários de maior porte. Em contrapartida, essa configuração aumenta a relevância da disponibilidade, da robustez e da proteção da entidade central de controle.

Configuração 3 – QKDN centralizada com nós hierárquicos: Na terceira configuração, a Y.3802 apresenta um modelo centralizado organizado hierarquicamente dos nós da QKDN, como apresentado da Figura 4.9. Para isso, a arquitetura passa a distinguir três tipos de nós conforme sua função predominante: *QKDN user node*, *QKDN access node* e *QKDN relay node* [ITU-T 2020c]. Essa diferenciação funcional é particularmente adequada a redes de maior escala e com cobertura geográfica mais ampla, nas quais os papéis de acesso, agregação e retransmissão tendem a se tornar mais especializados. Nesse arranjo, os nós de usuário conectam-se à rede para consumir o serviço de chaves, os nós de

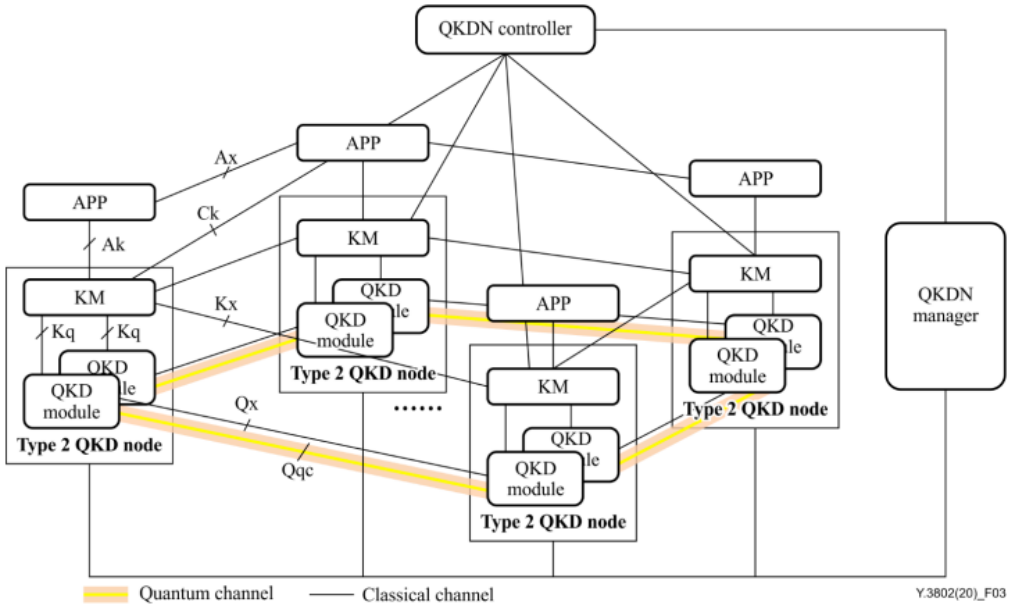


Figura 4.8. Configuração 2 da Y.3802, correspondente a uma QKDN centralizada, na qual as funções de controle são deslocadas para um ou mais controladores dedicados.

Fonte: Figura 3 ITU-T Y.3802 [ITU-T 2020c].

acesso atuam como pontos de entrada e agregação, e os nós de *relay* assumem papel mais direto na retransmissão de chaves entre diferentes segmentos da QKDN. Como resultado, a arquitetura torna-se mais compatível com cenários de implantação em larga escala, nos quais diferentes porções da rede exercem funções distintas e complementares.

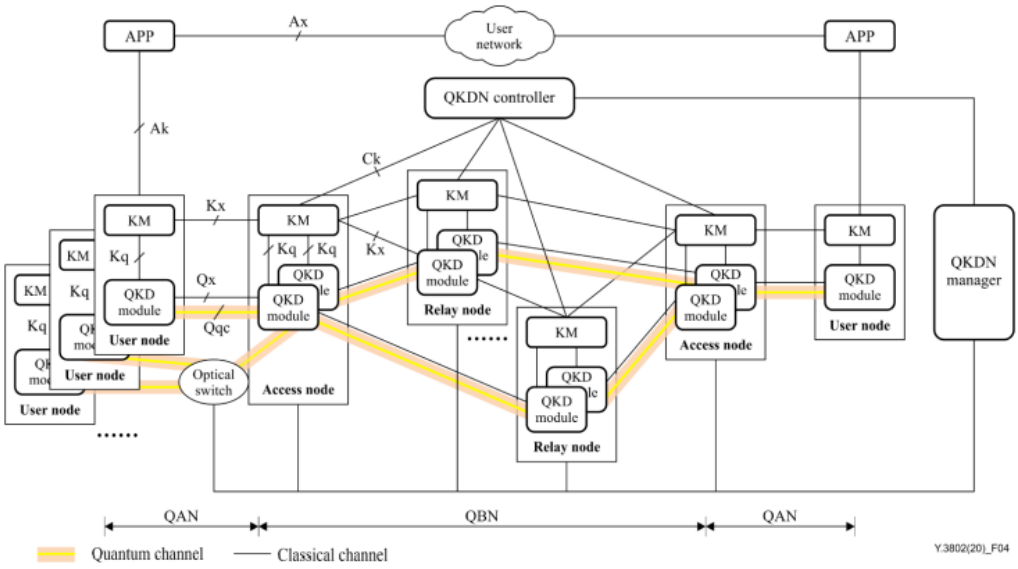


Figura 4.9. Configuração 3 da Y.3802, correspondente a uma QKDN centralizada com nós hierárquicos, distinguindo nós de usuário, de acesso e de *relay*.

Fonte: Figura 4 ITU-T Y.3802 [ITU-T 2020c].

Configuração 4 – QKDN centralizada com *relay* de chaves centralizado: Na quarta configuração, apresentada na Figura 4.10, a Y.3802 amplia o grau de centralização da arquitetura ao concentrar também a função de *relay* de chaves em uma entidade central, em vez de distribuí-la entre os nós da rede [ITU-T 2020c]. Com isso, torna-se mais nítida a separação entre as funções de geração de chaves, de gerenciamento local e de coordenação do *relay*. Essa organização pode ser especialmente vantajosa em cenários que exigem um controle global mais rigoroso do fluxo de chaves, seja para otimizar o uso de recursos, seja para aplicar políticas unificadas de *relay* ou reduzir a complexidade funcional dos nós. Em contrapartida, essa configuração aumenta a dependência de componentes centrais da arquitetura e, por isso, exige mecanismos mais robustos de gerenciamento, de proteção operacional e de tolerância a falhas.

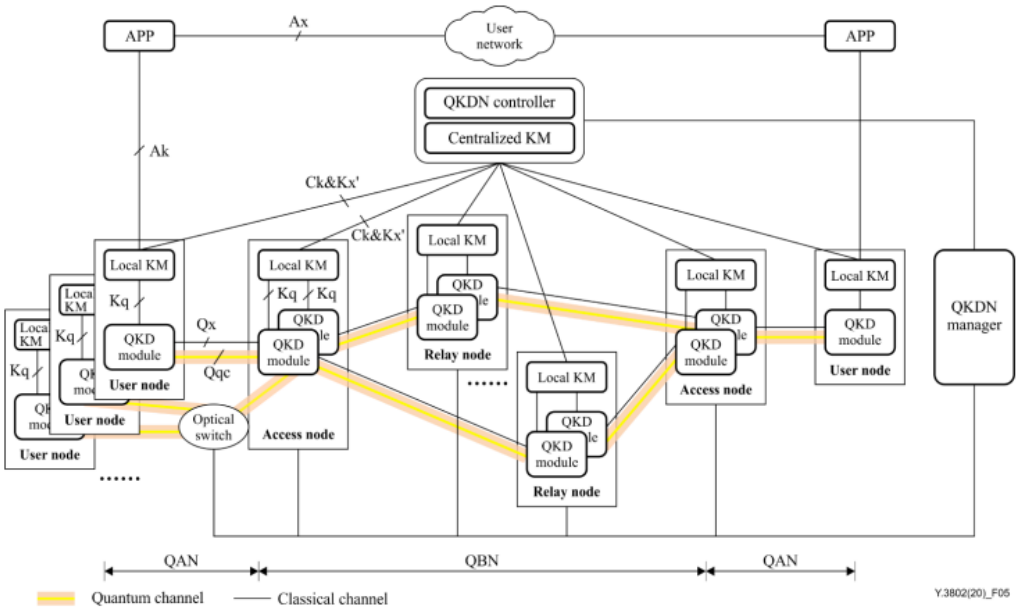


Figura 4.10. Configuração 4 da Y.3802, correspondente a uma QKDN centralizada com *relay* de chaves centralizado, na qual a coordenação do *relay* é concentrada em entidade dedicada.

Fonte: Figura 5 ITU-T Y.3802 [ITU-T 2020c].

4.4.6. Procedimentos operacionais básicos

A Y.3802 não se restringe à descrição estática dos componentes da QKDN. A recomendação também especifica procedimentos operacionais fundamentais para o funcionamento da rede, incluindo provisão de serviço e inicialização do sistema, geração de chaves, requisição e fornecimento de chaves, *relay* de chaves e reroteamento de *relay* [ITU-T 2020c]. Esse aspecto é particularmente relevante porque evidencia que a arquitetura funcional não deve ser compreendida apenas como um conjunto de entidades e interfaces, mas como um modelo normativo de operação, no qual o comportamento dinâmico da rede também é explicitamente previsto.

No procedimento de provisão e inicialização, a arquitetura prepara as entidades e os recursos necessários ao funcionamento da QKDN. Em seguida, no procedimento de geração de chaves, os módulos QKD produzem o material criptográfico e o transferem à

camada de gerenciamento de chaves. Já no procedimento de requisição e fornecimento, as aplicações criptográficas solicitam chaves e as recebem por meio das funções apropriadas da camada de *supply*. No procedimento de *relay*, por sua vez, a arquitetura viabiliza o compartilhamento de chaves entre nós que não possuem enlace direto. Por fim, no procedimento de reroteamento, a camada de controle ajusta os caminhos de *relay* diante de falhas, degradação de desempenho ou indisponibilidade de recursos, de modo a contornar enlaces ou nós problemáticos [ITU-T 2020c].

A contribuição central da Y.3802 é tornar explícito como a QKDN deve ser organizada para operar como rede. A recomendação transforma a estrutura conceitual da Y.3800 e os requisitos funcionais da Y.3801 em um arranjo coerente de camadas, entidades, interfaces, configurações e procedimentos. Nesse arranjo, o gerenciamento de chaves aparece como função estrutural, não apenas operacional, porque é ele que conecta a geração quântica de chaves ao *relay* entre nós, ao fornecimento para aplicações e à coordenação com as camadas de controle e gerenciamento. É por isso que a Y.3802 ocupa posição central na série: ela é a recomendação em que a QKDN passa a existir, de fato, como arquitetura funcional padronizada.

4.4.7. Gerenciamento de chaves na QKDN

A recomendação ITU-T Y.3803 especifica o gerenciamento de chaves em redes de distribuição de chaves quânticas. Seu escopo cobre a visão geral do gerenciamento de chaves, os elementos funcionais associados a essa atividade, as operações de gerenciamento e os formatos de chave utilizados na QKDN [ITU-T 2020d]. No encadeamento da série Y.3800, essa recomendação ocupa posição central porque traduz, em termos operacionais, a função que conecta a geração quântica de chaves ao seu uso por aplicações criptográficas em rede.

A própria Y.3803 afirma que o gerenciamento de chaves é a questão de maior prioridade para a operação eficiente e segura de uma QKDN, pois, sem ele, a maior parte das operações e serviços significativos da rede não pode ser realizada [ITU-T 2020d]. Essa observação é arquiteturalmente relevante. A utilidade prática da QKD em rede não decorre apenas da capacidade de módulos QKD produzirem cadeias simétricas de bits, mas da capacidade da rede de receber esse material, tratá-lo, armazená-lo, retransmiti-lo entre nós, sincronizá-lo entre extremidades e fornecê-lo de forma confiável às aplicações. Nesse sentido, a Y.3803 é a recomendação em que o KMS aparece de forma mais explícita como função estruturante da QKDN.

Modelo funcional interno da camada de gerenciamento de chaves: A Y.3803 não trata o *Key Manager* como uma entidade monolítica. Antes de detalhar operações como armazenamento, *relay* e fornecimento de chaves, a recomendação propõe uma decomposição funcional da camada de gerenciamento de chaves, ilustrada na Figura 4.11. Essa decomposição, alinhada à arquitetura funcional definida na Y.3802, torna explícita a separação entre três conjuntos de responsabilidades no interior do KM: as funções voltadas à gestão interna do material criptográfico, incluindo recebimento, organização, armazenamento e controle do ciclo de vida das chaves; as funções responsáveis pelo fornecimento de chaves às aplicações criptográficas; e as funções de comunicação do KM com as cama-

das de controle e de gerenciamento da QKDN, por meio das quais circulam informações operacionais, de coordenação e de supervisão [ITU-T 2020c, ITU-T 2020d].

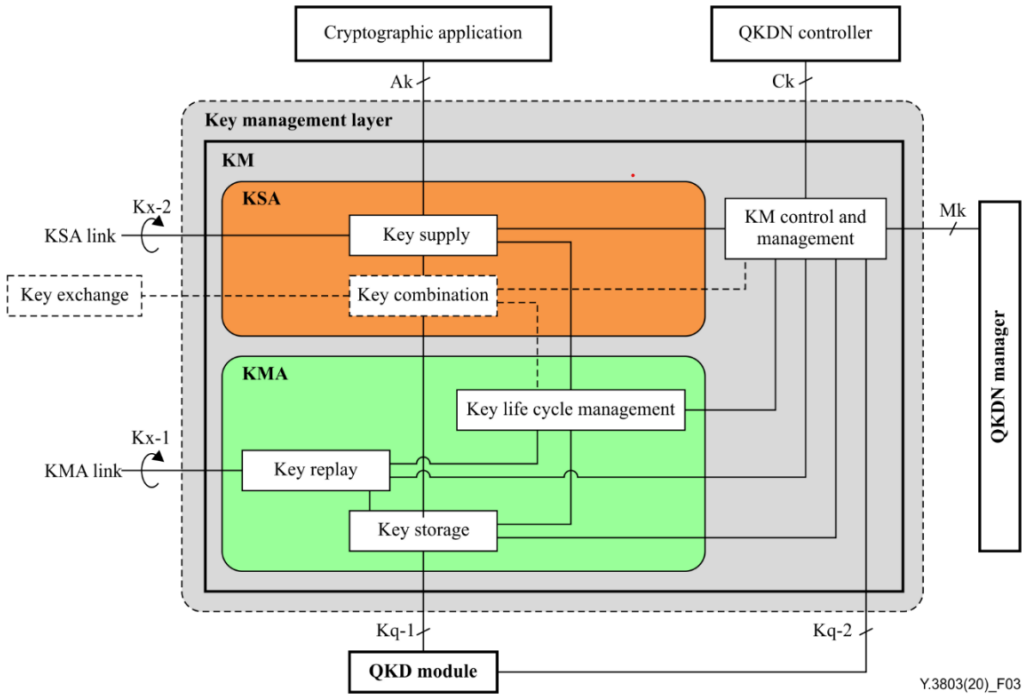


Figura 4.11. Modelo funcional da camada de gerenciamento de chaves na QKDN, com destaque para a decomposição do KM em KMA, KSA, função de controle e gerenciamento do KM, enlaces KMA/KSA e a fronteira de demarcação de segurança entre o KSA e a aplicação criptográfica.

Fonte: Figura 3 ITU-T Y.3803 [ITU-T 2020d].

Segundo a Y.3803, é conveniente identificar dois elementos funcionais no interior do KM: o *Key Management Agent* e o *Key Supply Agent* [ITU-T 2020d]. O KMA é responsável pela gestão interna das chaves no âmbito da QKDN. Para isso, recebe as chaves provenientes dos módulos QKD, participa da interconexão entre os nós da rede por meio de *key relay* e executa funções de armazenamento, retransmissão e gerenciamento do ciclo de vida das chaves. Assim, é no KMA que a chave deixa de ser tratada apenas como resultado do processo quântico de geração e passa a constituir um recurso gerenciado pela infraestrutura de rede [ITU-T 2020d].

O KSA, por sua vez, localiza-se entre o KMA e a aplicação criptográfica e realiza a função de *key supply*. A recomendação o descreve como a entidade que abriga interfaces para diferentes aplicações criptográficas e que pode, opcionalmente, incluir a função de *key combination*, na qual uma chave vinda do KMA é associada a outra chave proveniente de método externo de troca, preservando o nível de segurança da chave de entrada proveniente do KMA [ITU-T 2020d]. Essa possibilidade é relevante em cenários híbridos de integração entre QKD e outros mecanismos de distribuição de chaves.

Além das funções desempenhadas por KMA e KSA, o KM também incorpora a função de *KM Control and Management*, responsável pela articulação da camada de gerenciamento de chaves com os demais componentes da arquitetura da QKDN. Segundo a Y.3803, essa função estabelece a comunicação do KM com os módulos QKD, com o

controlador da QKDN e com o gerenciador da rede [ITU-T 2020d]. Sua relevância decorre do fato de que o gerenciamento de chaves não ocorre de forma isolada: ele depende do recebimento de informações da camada quântica, da disponibilização de informações operacionais às camadas superiores e da cooperação com funções de controle e gerenciamento associadas às decisões de *relay*, rerroteamento e supervisão da rede.

A norma Y.3803 também refina a noção de enlace do KM. Em vez de um único *KM link* indiferenciado, a recomendação estabelece que esse enlace é composto por um *KMA link* e um *KSA link* [ITU-T 2020d]. O KMA link conecta KMAs para realizar *relay* e outras comunicações de gerenciamento. O *KSA link* conecta KSAs para sincronização e verificação de integridade antes do fornecimento da chave à aplicação. Essa separação ajuda a distinguir o domínio do gerenciamento interno do material criptográfico do domínio de sua preparação final para consumo no plano de serviço.

A Figura 4.11 também destaca um aspecto arquitetural fundamental: a *security demarcation boundary* estabelecida entre o KSA e a aplicação criptográfica. Essa fronteira define o ponto em que a responsabilidade pela chave deixa de pertencer à infraestrutura da QKDN e passa a ser atribuída à aplicação que a utiliza [ITU-T 2020d]. Até esse limite, a chave permanece sujeita às políticas de armazenamento, sincronização, rastreabilidade e eliminação estabelecidas pelo gerenciamento da QKDN. A partir do *key supply*, entretanto, seu uso passa a ser regido pela lógica operacional da aplicação consumidora.

4.4.8. Operações de gerenciamento de chaves

Uma vez definido o modelo funcional interno do KM, a norma Y.3803 passa a descrever as operações que compõem o gerenciamento de chaves. A recomendação organiza essas operações em torno da trajetória da chave desde sua geração na camada quântica até seu uso pela aplicação criptográfica, abrangendo aquisição, autenticação, armazenamento, *relay*, fornecimento, reroteamento e gerenciamento do ciclo de vida [ITU-T 2020d]. A Figura 4.12 sintetiza as principais operações de gerenciamento e as interações entre KMA, KSA, os módulos QKD, o controlador, o gerenciador da rede e a aplicação criptográfica. Essa sequência de operações transforma o modelo funcional da Figura 4.11 em um fluxo operacional de rede.

Aquisição, autenticação, armazenamento, formatação e metadados: A Y.3803 trata a aquisição e o armazenamento de chaves como um processo estruturado, e não como simples gravação local de sequências de bits. Na camada quântica, cada par de módulos QKD gera uma *QKD-key*, entendida como a sequência simétrica de bits antes de qualquer redimensionamento ou formatação no *Key Manager*. Como diferentes pares de módulos QKD podem empregar protocolos distintos e produzir unidades de chave com comprimentos diferentes, a recomendação estabelece que, já no módulo QKD, os metadados sejam anexados à *QKD-key*, formando um *key file*, que é então transferido ao KMA correspondente [ITU-T 2020d].

No KMA, a primeira etapa é receber os *QKD-key files* do ponto de referência *Kq-1*. Como os comprimentos dessas chaves podem variar, a Y.3803 recomenda que o KMA padronize as *QKD-keys*, combinando ou dividindo os bit *strings* em chaves de comprimento unitário prescrito, adequado à política interna de gerenciamento e ao uso posterior

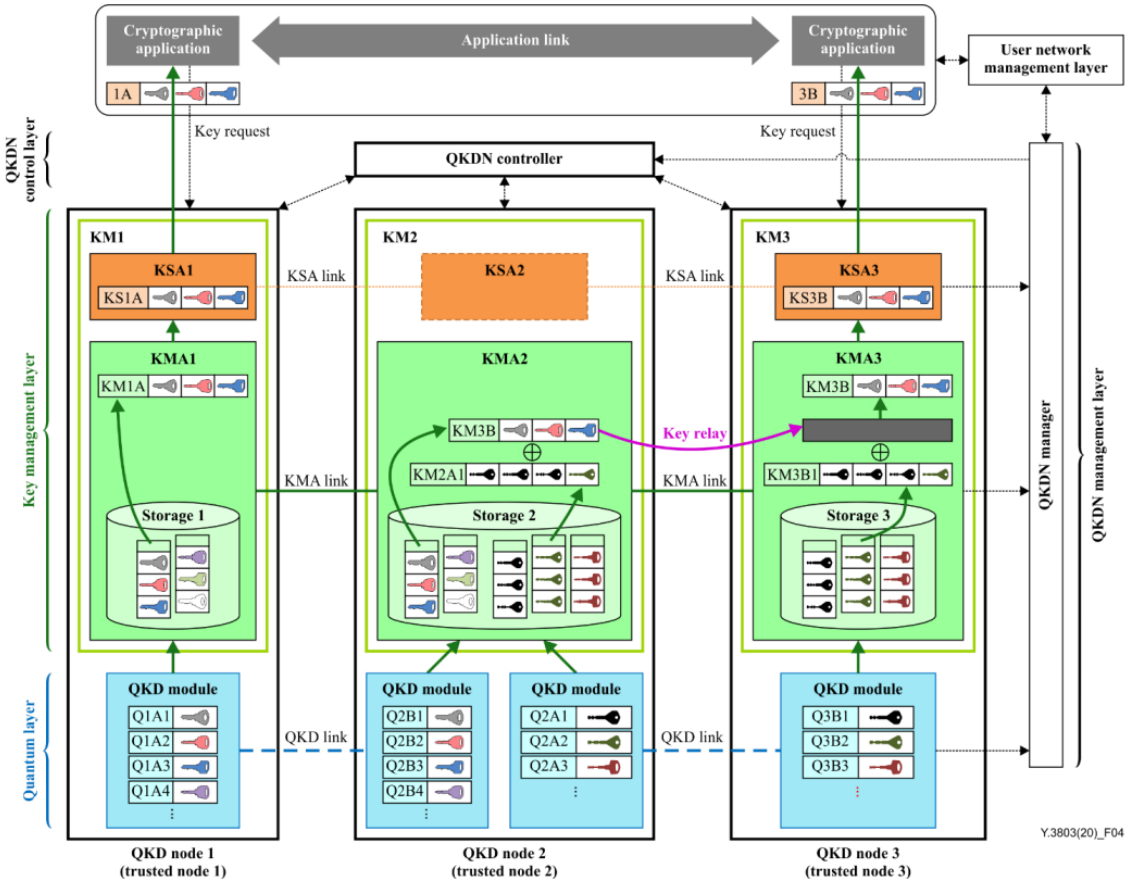


Figura 4.12. Principais operações de gerenciamento de chaves em uma QKDN. A figura ilustra o percurso da chave desde sua geração na camada quântica, passando pela aquisição pelo KMA, pelas funções de *relay* e sincronização, até sua entrega (*key supply*), em articulação com funções de controle e gerenciamento da rede.

Fonte: Figura 4 ITU-T Y.3803 [ITU-T 2020d].

na rede. Antes do armazenamento definitivo, essas chaves formatadas são mantidas temporariamente em um *buffer* [ITU-T 2020d]. Esse detalhe é importante, pois mostra que a formatação não é uma operação opcional, mas parte constitutiva da transição da chave do domínio de geração quântica para o domínio de gerenciamento de rede.

A recomendação também exige que o armazenamento definitivo seja precedido da verificação da identidade da chave entre os KMAs correspondentes. Assim, os KMAs autenticam-se mutuamente por meio do *KMA link*, e um deles envia ao outro uma requisição de autenticação contendo o identificador da *QKD-key* e um valor de verificação, como um *hash* criptográfico ou um *message authentication code*. O KMA par, então, realiza sincronização em posição de bits e autenticação do conteúdo armazenado em *buffer*, comparando o valor recebido com o calculado localmente. Se os valores coincidirem, as chaves *buffered* são finalmente armazenadas como *KMA-key*, acompanhadas de metadados no diretório de armazenamento; caso contrário, o material é abortado [ITU-T 2020d].

Esse ponto é central para compreender a diferença entre *QKD-key* e *KMA-key*. A primeira corresponde ao material oriundo diretamente dos módulos QKD. A segunda

corresponde à chave já admitida no domínio de gerenciamento do KMA, após recepção, reformatação, sincronização e autenticação. Portanto, o KMA não apenas guarda a chave, mas a transforma em um objeto gerenciável pela arquitetura da QKDN [ITU-T 2020d].

A Y.3803 também formaliza a noção de *key file* como o conjunto lógico composto por dados-chave e metadados. Essa estrutura é necessária para manter a interconectividade e a expansibilidade na QKDN, pois os metadados não servem apenas para identificação local. Eles também sustentam comunicações entre KMAs, KSAs, aplicações criptográficas, controlador e gerenciador da rede [ITU-T 2020d]. Para a *QKD-key*, a recomendação inclui, como metadados básicos, o identificador da chave, além de campos opcionais, como o comprimento, o identificador do módulo QKD, o identificador do módulo correspondente, o *timestamp* de geração e o *hash* da chave. Para a *KMA-key*, são previstos, entre outros, identificador único na QKDN, comprimento, tipo da chave, identificador do KMA, identificador do KMA correspondente, *timestamp* de geração, referência ao módulo QKD de origem e *hash* da chave [ITU-T 2020d].

Do ponto de vista arquitetural, isso significa que o armazenamento de chaves na Y.3803 é inseparável da formatação e dos metadados. A chave não é mantida apenas como sequência de bits secretos, mas também como entidade identificável, sincronizável e rastreável, apta a participar de operações posteriores de *relay*, *supply* e de gerenciamento do ciclo de vida. É essa estrutura que permite ao KMS operar de forma interoperável, auditar a trajetória da chave e oferecer suporte às camadas de controle e gerenciamento da QKDN.

Relay entre KMAs: Quando dois nós não possuem enlace QKD direto, a QKDN depende do *key relay* para permitir o compartilhamento de chaves entre extremos arbitrários. A Y.3803 trata essa operação como função central do KMA, realizada por meio de KMA links entre nós confiáveis [ITU-T 2020d]. Do ponto de vista arquitetural, isso significa que o KMA não apenas armazena chaves localmente, mas também participa da extensão lógica da rede ao retransmitir material criptográfico entre segmentos distintos da QKDN.

O *relay* também evidencia a interdependência entre Y.3803 e Y.3804. Embora o conteúdo criptográfico permaneça sob responsabilidade do gerenciamento de chaves, a viabilidade do *relay* depende do suporte do controlador da QKDN para a definição e a manutenção das rotas apropriadas. Assim, a Y.3803 trata o *relay* como uma operação de gerenciamento de chaves, mas essa operação é sustentada por decisões de controle e de gerenciamento de rede [ITU-T 2020d, ITU-T 2020b]. Essas decisões podem incluir o re-roteamento do *relay* entre KMAs quando a rota inicialmente escolhida deixa de ser operacionalmente adequada. A recomendação indica que isso pode ocorrer, por exemplo, quando a quantidade de chaves disponíveis em um nó intermediário fica abaixo de um limiar, quando há falha em KM ou em *KMA link*, ou quando um aumento de QBER em determinado enlace QKD reduz ou impede a geração de chaves necessárias para sustentar a rota em uso [ITU-T 2020d]. Nesses casos, a camada de gerenciamento de chaves precisa cooperar com o controlador e com o gerenciamento da QKDN para reconfigurar o caminho lógico do *relay* e preservar a continuidade do serviço de chaves.

Fornecimento e ciclo de vida de chaves: Se, por um lado, o *relay* amplia a conectividade da QKDN, por outro, o *key supply* materializa sua utilidade como serviço. A recomendação define essa operação como a entrega da chave à aplicação criptográfica por meio do KSA [ITU-T 2020d]. Para isso, o KSA sincroniza as chaves entre as extremidades, verifica a integridade via *KSA link* e então as fornece à aplicação pela interface apropriada. É nesse ponto que a QKDN se apresenta ao plano de serviço não mais como rede interna de gerenciamento, mas como provedora de recurso criptográfico utilizável por protocolos, sistemas e aplicações.

Por fim, a recomendação integra todas essas operações ao conceito de ciclo de vida da chave. O *Key Life Cycle* é definido como a sequência de etapas que a chave percorre desde sua recepção pelo KM, passando por armazenamento, formatação, *relay*, sincronização, autenticação e *supply*, até sua eliminação ou preservação conforme a política vigente [ITU-T 2020d]. Essa formulação reforça que o gerenciamento de chaves em uma QKDN não é apenas manipulação pontual de bit strings, mas governança contínua de um recurso criptográfico distribuído.

Síntese da contribuição da Y.3803: A organização da subseção em dois níveis, modelo funcional e operações, reflete a própria lógica da Y.3803. Primeiro, a recomendação mostra como o KM é decomposto em KMA, KSA, função de controle e enlaces específicos. Depois, mostra como esses elementos executam aquisição, armazenamento, *relay*, *supply*, rerroteamento e gerenciamento do ciclo de vida [ITU-T 2020d]. Essa sequência é importante porque permite compreender por que o KMS não deve ser interpretado como um simples repositório de chaves, mas como uma função estrutural da rede, responsável por transformar a geração quântica de bits em provisão segura, rastreável e interoperável de chaves para aplicações distribuídas.

4.4.9. Controle e gerenciamento da QKDN:

A recomendação ITU-T Y.3804 especifica as funções e os procedimentos de controle e gerenciamento das redes de distribuição de chaves quânticas. Seu escopo cobre os elementos funcionais de controle, gerenciamento e orquestração da QKDN, as funções associadas a esses elementos e os procedimentos necessários para sustentar a operação da rede [ITU-T 2020b]. No encadeamento da série Y.3800, essa recomendação complementa a arquitetura funcional definida em Y.3802 ao explicitar como a rede deve ser supervisionada, coordenada e mantida em operação diante de falhas, degradação de desempenho, variações de disponibilidade e requisitos de segurança.

Do ponto de vista arquitetural, a Y.3804 é importante porque consolida a QKDN como uma infraestrutura de telecomunicações gerenciável. Se a Y.3803 mostra como o material criptográfico é armazenado, sincronizado, retransmitido e fornecido às aplicações, a Y.3804 mostra como a rede que sustenta essas operações é controlada e gerenciada como um todo. Portanto, a recomendação desloca a discussão do domínio do fluxo de chaves para o domínio da governança operacional da infraestrutura que produz, distribui e entrega essas chaves.

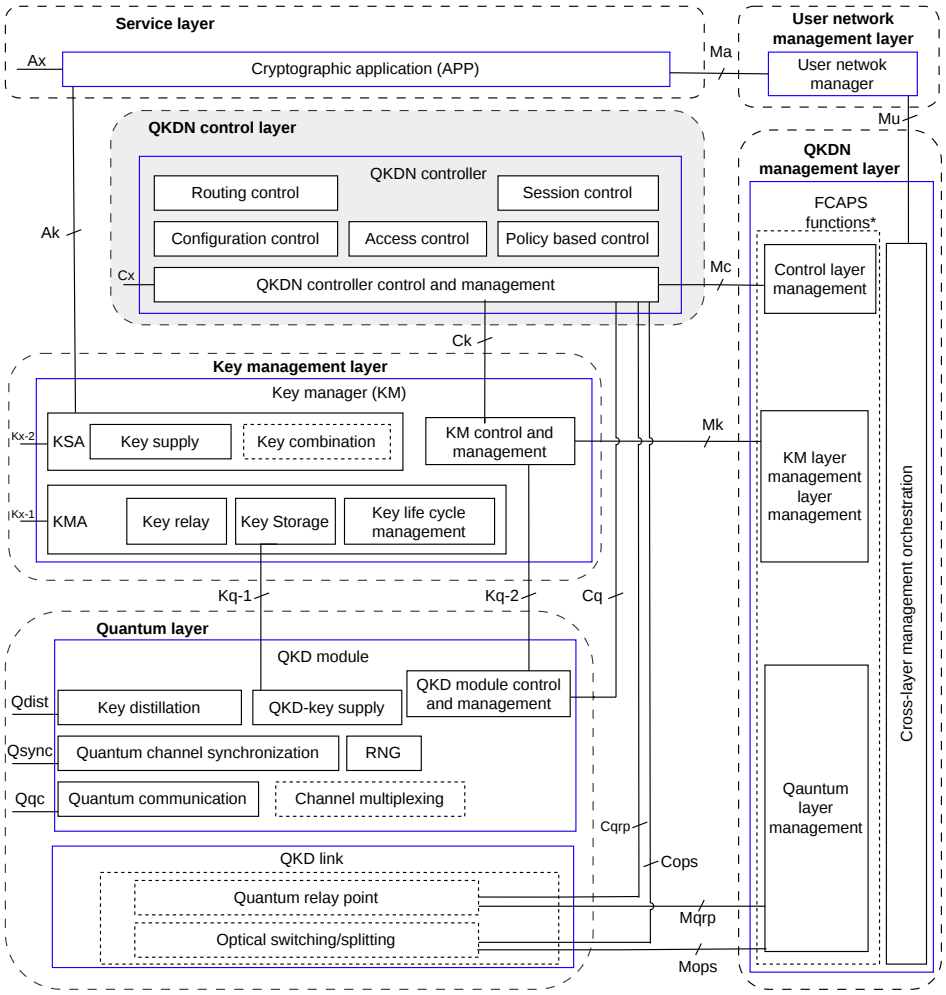


Figura 4.13. Elementos funcionais e pontos de referência relevantes para o controle e o gerenciamento da QKDN.

Fonte: Figura 1 ITU-T Y.3804 [ITU-T 2020b].

Arquitetura funcional de controle e gerenciamento: A Y.3804 organiza o problema em torno de duas esferas funcionais: a camada de controle da QKDN e a camada de gerenciamento da QKDN [ITU-T 2020b]. A camada de controle atua sobre os recursos da rede para garantir operação segura, estável, eficiente e robusta. A camada de gerenciamento, por sua vez, supervisiona a rede como um todo, coleta informações de múltiplas camadas, sustenta decisões operacionais e provê mecanismos de orquestração entre diferentes domínios funcionais.

A Figura 4.13 deve ser posicionada nesta subseção, pois sintetiza os elementos funcionais e os pontos de referência mais relevantes para o controle e o gerenciamento da QKDN. Em especial, ela evidencia que a recomendação não trata apenas de monitoramento passivo, mas de uma estrutura explícita de interação entre controlador, gerenciador, camada quântica, camada de gerenciamento de chaves e rede do usuário.

A recomendação define, para fins de controle, pontos de referência como *Cx*, *Ck*,

Cq, *Cops* e *Cqrp*, empregados na comunicação entre o controlador da QKDN e os componentes sob controle. Para fins de gerenciamento, define pontos de referência como *Mq*, *Mqrp*, *Mops*, *Mk*, *Mc*, *Mx* e *Mu*, empregados na comunicação entre o gerenciador da QKDN e os elementos sob gerenciamento [ITU-T 2020b]. Essa explicitação de interfaces é importante porque transforma o controle e o gerenciamento em funções arquiteturalmente observáveis, integráveis e interoperáveis.

Funções da camada de controle: A Y.3804 atribui à camada de controle cinco funções principais: controle de roteamento, controle de configuração, controle baseado em políticas, controle de acesso e controle de sessão [ITU-T 2020b]. Em conjunto, essas funções permitem que a rede ajuste dinamicamente o uso de recursos e mantenha a continuidade do serviço de chaves.

A função de roteamento é especialmente relevante em QKDNs baseadas em *trusted nodes*, pois é responsável por provisionar rotas apropriadas de *key relay* entre os extremos e por rerroteá-las quando a camada quântica ou a camada de gerenciamento de chaves indica degradação ou indisponibilidade [ITU-T 2020b]. Para isso, o controlador mantém informações de endereçamento de nós, identifica KMs envolvidos, consulta o consumo e o estoque residual de chaves, recebe parâmetros dos enlaces QKD e combina essas informações com a topologia global da rede. O resultado é uma decisão de *relay* que não depende apenas da conectividade física, mas também da disponibilidade efetiva de recursos criptográficos ao longo do caminho.

A função de configuração controla o estado de módulos e enlaces, incluindo a ativação, a desativação, a reserva e a reconfiguração. A recomendação relaciona esse ponto a situações em que falhas ou alterações de condição, como o aumento de QBER ou a perda de enlace, exigem a recomposição da infraestrutura operacional [ITU-T 2020b]. A função baseada em políticas, por sua vez, associa o uso da rede a critérios como QoS, prioridades e cobrança. Já a função de controle de acesso estabelece mecanismos de autenticação e autorização para componentes da rede, o que é especialmente importante em uma arquitetura cujo domínio de confiança depende de uma coordenação rigorosa entre nós, módulos QKD, KMs e aplicações. Por fim, a função de controle de sessão apoia o estabelecimento de fluxos de *relay* e de *supply*, articulando-se ao gerenciamento de chaves e às políticas da rede [ITU-T 2020b].

Um ponto conceitualmente importante da recomendação é a afirmação explícita de que o controlador da QKDN **não lida diretamente com as chaves**. As chaves são fornecidas do KM à aplicação criptográfica, enquanto o controlador estabelece as condições para que o *relay* e o *supply* ocorram de forma contínua e com coerência operacional [ITU-T 2020b]. Essa distinção preserva a separação arquitetural entre o domínio do material criptográfico e o da coordenação da rede.

Funções da camada de gerenciamento: Na camada de gerenciamento, a Y.3804 estrutura as funções segundo a lógica FCAPS, cobrindo falhas, configuração, contabilidade, desempenho e segurança [ITU-T 2020b]. Isso coloca a QKDN em continuidade com modelos já consolidados de gestão de redes de telecomunicações, mas adaptando-os às

especificidades da distribuição quântica de chaves.

No gerenciamento de falhas, o sistema monitora eventos e apoia diagnósticos e ações corretivas relacionados a módulos QKD, enlaces QKD, KMs e enlaces de gerenciamento. No gerenciamento de configuração, mantém a topologia, o inventário de recursos e os estados operacionais. No gerenciamento de contabilidade, mede o uso de recursos e apoia políticas de cobrança. No gerenciamento de desempenho, coleta e analisa dados operacionais da rede. No gerenciamento de segurança, reúne informações relacionadas a eventos, registros, rastros de auditoria e suporte ao ciclo de vida das chaves [ITU-T 2020b].

A recomendação também explicita as funções específicas de cada camada. O gerenciamento da camada quântica acompanha, por exemplo, parâmetros de desempenho e detecção de tentativas de ataque ao canal quântico. O gerenciamento da camada de chaves acompanha o estoque e a disponibilidade de chaves para *relay* e *supply*. O gerenciamento da camada de controle apoia o controlador nas decisões de roteamento e de rerroteamento. Essa separação é importante porque evidencia que a camada de gerenciamento não atua como uma entidade abstrata única, mas como um mecanismo de supervisão articulado às necessidades específicas de cada camada da QKDN [ITU-T 2020b].

Orquestração entre camadas: Um aspecto relevante da Y.3804 é a introdução explícita de funções de orquestração entre camadas, reunidas sob a lógica de *cross-layer management and orchestration (XLMO)* [ITU-T 2020b]. Essa função coordena informações e ações entre a camada quântica, a camada de gerenciamento de chaves e a camada de controle, além de interagir com sistemas externos de gerenciamento, especialmente o gerenciamento da rede do usuário.

Essa orquestração é importante porque a QKDN não opera como um conjunto de camadas independentes. Um aumento de QBER na camada quântica pode exigir reroteamento decidido pelo controlador. A redução do estoque de chaves em nós intermediários pode afetar a viabilidade de determinadas rotas de *relay*. Uma mudança na política de serviço ou na prioridade de aplicação pode afetar as decisões de *supply*. Em todos esses casos, a operação coerente da rede depende de coordenação transversal. A XLMO formaliza exatamente essa necessidade, transformando-a em uma função arquitetural explícita [ITU-T 2020b].

Além de definir funções de controle e gerenciamento, a ITU-T Y.3804 explicita procedimentos operacionais que mostram como essas funções interagem em situações concretas de operação da QKDN, incluindo falhas, configuração, desempenho, segurança, *key relay* e reroteamento de *key relay* [ITU-T 2020b]. No contexto deste capítulo, os procedimentos de *relay* e de reroteamento são relevantes porque evidenciam, de forma dinâmica, a divisão de responsabilidades entre a camada de gerenciamento de chaves e as camadas de controle e de gerenciamento da rede.

O ponto de partida desses procedimentos é a separação arquitetural já estabelecida nas recomendações anteriores. A Y.3802 indica que o *Key Manager* executa o *relay* quando necessário, mas pode depender do controlador da QKDN para obter a rota apropriada [ITU-T 2020c]. A Y.3803, por sua vez, esclarece que a continuidade do *relay* de-

pende do estado das chaves disponíveis nos nós intermediários e das condições da camada quântica, como alarmes e degradação associados ao aumento de QBER [ITU-T 2020d]. A Y.3804 formaliza essa dinâmica e mostra, passo a passo, como a rede transforma estado operacional em ação de controle.

Procedimento de *key relay*: O procedimento de *key relay* definido na Y.3804, ilustrado na Figura 4.14, descreve o fluxo pelo qual um KM obtém, por meio do controlador da QKDN, a rota lógica necessária para compartilhar chaves com outro KM não adjacente [ITU-T 2020b]. O procedimento começa quando um KM solicita uma rota de *relay* à função de controle de sessão (*session control function*) do controlador da QKDN. Essa escolha é significativa porque mostra que o *relay* não é tratado apenas como uma decisão de roteamento isolada, mas como parte do gerenciamento de uma sessão lógica de compartilhamento de chaves entre um nó de origem e um nó de destino.

Recebida a solicitação, a função de controle de sessão verifica inicialmente se já existe uma sessão estabelecida entre o nó QKD de origem e o nó QKD de destino. Caso essa sessão ainda não exista, a função de controle de sessão consulta a função de controle de roteamento (*routing control function*) para determinar uma rota adequada para *relay*. Em seguida, a função de roteamento consulta a função de controle de configuração (*configuration control function*), da qual obtém as informações de configuração da rede necessárias para avaliar os caminhos possíveis. Com base nessas informações, a função de roteamento calcula uma rota ótima para o *relay* e a devolve à função de controle de sessão, que então a repassa ao KM solicitante [ITU-T 2020b].

Do ponto de vista funcional, esse procedimento evidencia três aspectos centrais. Primeiro, a decisão de *relay* não é tomada localmente pelo KM com base apenas na conectividade imediata, mas depende de uma visão lógica da rede mediada pelo controlador. Segundo, a determinação da rota não se reduz a um problema abstrato de grafo, pois depende de informações sobre a configuração efetiva da infraestrutura. Terceiro, o resultado do procedimento não é a própria chave, mas sim a informação de rota que habilita a camada de gerenciamento de chaves a executar o *relay*. Em outras palavras, o controlador não manipula diretamente o material criptográfico, mas fornece ao KM as condições operacionais para que o *relay* ocorra de forma coerente com o estado da rede.

Essa leitura é compatível com a descrição geral da função de roteamento na própria Y.3804, segundo a qual o controlador deve gerenciar a tabela de rotas, adquirir informações sobre o consumo de chaves, o estoque residual nos KMs, os parâmetros dos enlaces QKD e a topologia da rede, e então provisionar uma rota apropriada entre os KMs extremos [ITU-T 2020b]. A Figura 4.14 traduz essa função em um encadeamento operacional mais concreto.

Procedimento de rerroteamento de *key relay*: Se o procedimento anterior trata da provisão inicial de uma rota de *relay*, o procedimento de rerroteamento, ilustrado na Figura 4.15, trata da adaptação dessa rota quando a continuidade do serviço de chaves está ameaçada [ITU-T 2020b]. Aqui, a ênfase da recomendação desloca-se da simples decisão de caminho para a cooperação entre gerenciamento e controle diante de mudanças de

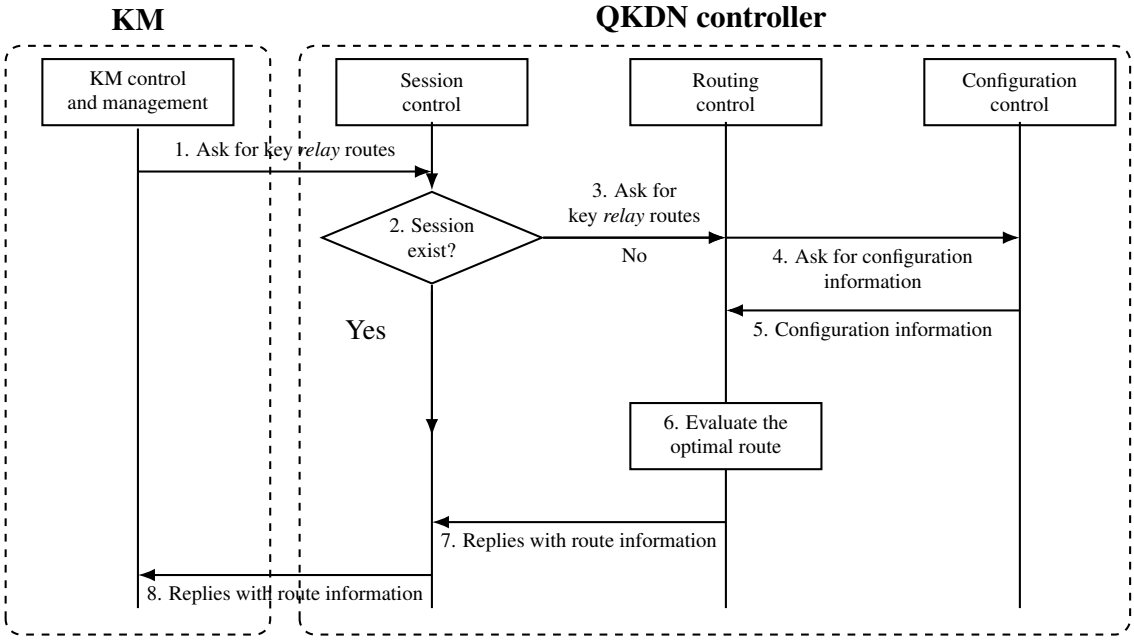


Figura 4.14. Procedimento de *key relay* em uma QKDN, adaptado da Figure 14 da ITU-T Y.3804. A figura mostra como a requisição de *relay* parte do KM, é tratada pelas funções de sessão, roteamento e configuração do controlador da QKDN, e retorna ao KM como informação de rota para execução do *relay*.

estado na rede.

Segundo a Y.3804, o processo começa quando a função de gerenciamento da camada de controle da QKDN (*QKDN Control Layer Management*, QCLM) recebe, do controlador e de suas funções de controle e gerenciamento, uma requisição de informações de suporte para roteamento ou rerroteamento de *relay*. A QCLM, então, solicita à função de orquestração e gerenciamento entre camadas (*Cross-Layer Management and Orchestration*, XLMO) informações sobre a topologia da rede. A XLMO devolve essas informações à QCLM, que passa a analisar a topologia e a selecionar as informações de *relay* e de suporte compatíveis com a situação observada. Por fim, a QCLM envia ao controlador da QKDN o conjunto de informações resultante, permitindo que este execute ações de roteamento ou rerroteamento do *relay* [ITU-T 2020b].

Esse procedimento é arquiteturalmente importante porque evidencia que o rerroteamento não é apresentado como uma reação puramente local do controlador. Ao contrário, a decisão depende de suporte explícito da camada de gerenciamento, especialmente da correlação entre informações topológicas e de estado realizada pela XLMO e processada pela QCLM. Assim, o rerroteamento aparece como produto da coordenação entre camadas, e não apenas como uma atualização da tabela de rotas.

A motivação desse procedimento torna-se mais evidente quando analisado em conjunto com as Recomendações Y.3803 e Y.3804. A Y.3803 indica que o rerroteamento do *relay* pode ser necessário quando o número de chaves disponíveis em nós intermediários cai abaixo de um determinado limiar, quando ocorrem falhas em KM ou no enlace de gerenciamento, ou ainda quando um QBER elevado compromete ou reduz a geração de chaves em um enlace QKD [ITU-T 2020d]. Em consonância com isso, a Y.3804, ao

descrever a função de roteamento, destaca que o rerroteamento pode ser acionado em dois conjuntos mais amplos de situações: aquelas associadas à camada de gerenciamento de chaves, como esgotamento de estoque ou falhas em KM/KM link, e aquelas relacionadas à camada quântica, como aumento do QBER ou falha de um módulo QKD [ITU-T 2020b]. Nesse sentido, o procedimento apresentado na Figura 4.15 explicita como essas informações são encaminhadas ao plano de gerenciamento e, posteriormente, retornam ao controlador como subsídio para a decisão sobre uma nova rota.

Em termos operacionais, o procedimento de rerroteamento aponta que a disponibilidade de serviço em uma QKDN depende de monitoramento contínuo e de uma cadeia de decisão que articula quatro níveis: (i) os elementos que detectam degradação ou falha; (ii) o gerenciamento por camada, que consolida essas informações; (iii) a orquestração entre camadas, que correlaciona topologia e estado global; e (iv) o controlador, que transforma esse suporte em ação sobre as rotas de *relay*. O serviço de chaves, portanto, não depende apenas da existência física de enlaces QKD, mas também da capacidade da rede de reconfigurar, de forma lógica, o uso desses recursos frente a eventos operacionais.

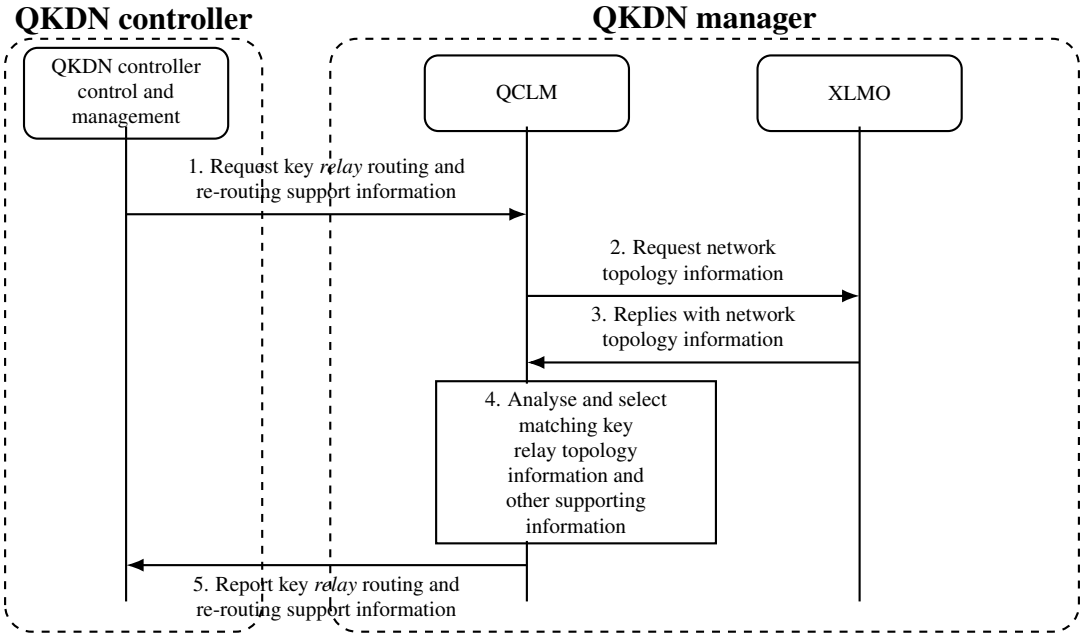


Figura 4.15. Procedimento de rerroteamento de *key relay* em uma QKDN, adaptado da Figure 15 da ITU-T Y.3804. A figura mostra como a QCLM e a XLMO fornecem ao controlador da QKDN o suporte topológico e operacional necessário para reconfiguração da rota de *relay*.

Interpretação conjunta dos dois procedimentos: As Figuras 4.14 e 4.15 devem ser lidas em conjunto. A primeira descreve a lógica de provisão inicial de uma rota de *relay*: uma solicitação parte do KM, percorre as funções de sessão, roteamento e configuração do controlador, e retorna ao KM como informação de rota. A segunda descreve a lógica de adaptação dessa rota quando o estado da rede torna necessário alterar o caminho previamente adotado: a demanda de suporte sobe da camada de controle ao gerenciamento por camada e à orquestração entre camadas, e então retorna ao controlador como base para

uma nova decisão de *relay* [ITU-T 2020b].

Em termos de controle e gerenciamento, essa distinção mostra-se bastante nítida. O *key relay* refere-se ao estabelecimento de um caminho lógico para o compartilhamento de chaves entre KMs não adjacentes, enquanto o *key relay rerouting* diz respeito à preservação desse serviço quando as condições operacionais deixam de sustentar a rota originalmente utilizada. Assim, o primeiro está associado à provisão do serviço, ao passo que o segundo se relaciona à sua resiliência. Em ambos os casos, a arquitetura padronizada da série Y.3800 mantém a mesma lógica de separação funcional: o material criptográfico permanece sob responsabilidade do gerenciamento de chaves, enquanto o controlador e as funções de gerenciamento da rede fornecem as condições estruturais

***Síntese da contribuição da Y.3804*:** A Y.3804 completa a base arquitetural da série Y.3800 ao demonstrar como a QKDN deve ser controlada, monitorada e orquestrada em operação. Se a Y.3803 torna o gerenciamento de chaves explicitamente central para a utilidade da rede, a Y.3804 mostra como essa utilidade depende de mecanismos de coordenação da infraestrutura: roteamento, reroteamento, controle de acesso, aplicação de políticas, supervisão FCAPS e articulação entre camadas [ITU-T 2020b].

Para este capítulo, a contribuição principal é tornar visível que uma QKDN não pode ser concebida apenas como um conjunto de módulos QKD e KMs. A rede também exige uma malha de controle e gerenciamento que acompanhe o estado da camada quântica, a disponibilidade do material criptográfico, a topologia operacional e as demandas das aplicações. É essa combinação entre geração quântica, gerenciamento de chaves e governança operacional da infraestrutura que transforma a QKD em um serviço de rede sustentado efetivamente.

4.4.10. Casos de uso representativos

O ITU-T Y Suppl. 80 é um documento complementar, de caráter informativo, da série Y.3800, que reúne casos de uso de QKDN no contexto das tecnologias de rede tratadas pelo SG13[4]. Esses casos são organizados em seis grandes grupos: combinação de QKD com primitivas criptográficas, integração com protocolos TCP/IP, implementação em diferentes topologias de rede, uso com distintas categorias de dispositivos, integração em diferentes tipos de rede e aplicação em setores verticais [ITU 2023]. Seu propósito é evidenciar que a arquitetura padronizada proposta pela ITU-T foi concebida para dar suporte a cenários concretos de integração com protocolos, topologias e serviços de rede. Em todos esses contextos, a existência de um sistema de gerenciamento de chaves funcionalmente bem definido constitui um requisito essencial para que a QKD possa ser oferecida como um serviço de rede interoperável e escalável.

Entre esses grupos, três são particularmente relevantes para a discussão de KMS e arquitetura de rede. O primeiro é a integração com protocolos TCP/IP, na qual o suplemento aponta o uso de QKD em camadas distintas da pilha, incluindo PPP e MACsec na camada de enlace, IPsec na camada de rede, TLS na camada de transporte e aplicações

[4]SG13 (*Study Group 13*) da ITU-T é responsável por estudos sobre arquitetura e serviços de redes futuras.

de camada superior. O segundo é o grupo de topologias de rede, que inclui explicitamente redes metropolitanas, *backbones* interurbanos e redes satélite-solo, o que evidencia que a QKDN deve ser tratada como infraestrutura de rede e não apenas como um enlace isolado. O terceiro é a integração em diferentes formas de rede, incluindo 4G/5G, Software Defined Networking (SDN)/Network Functions Virtualization (NFV), computação em nuvem, *blockchain* e outras arquiteturas futuras, o que reforça a necessidade de interfaces padronizadas de gerenciamento, controle e provisão de chaves [ITU 2023].

Entre as topologias destacadas no ITU-T Y Suppl. 80, a rede metropolitana de acesso ocupa posição relevante por representar um cenário de implantação próximo das infraestruturas urbanas de telecomunicações. O suplemento descreve esse caso de uso como uma rede de alta segurança para comunicações entre diferentes unidades ou escritórios em uma área metropolitana, tipicamente da ordem de 100 km de diâmetro, interconectada por enlaces ópticos dedicados para comunicação clássica e QKD. Nesse arranjo, as chaves geradas podem ser utilizadas tanto em mecanismos de autenticação quanto em primitivas de cifra, e, quando necessário, a conectividade, a disponibilidade e a largura de banda podem ser ampliadas com soluções baseadas em repetição confiável [ITU 2023].

Um exemplo representativo desse tipo de implantação é a *Madrid Quantum Network*. Trata-se de uma rede metropolitana baseada em fibra óptica, composta por 12 nós distribuídos entre centros de pesquisa, empresas e universidades, e integrada à infraestrutura de telecomunicações da Telefónica e da RediMadrid [García Cid et al. 2021]. Segundo os resultados reportados, o maior enlace da rede possui aproximadamente 55 km, a taxa máxima de geração de chaves atinge 70 kbps em alguns trechos e as perdas variam de cerca de 0,20 dB/km a 1,1 dB/km em segmentos mais degradados da região central urbana [García Cid et al. 2021]. Do ponto de vista arquitetural, a rede de Madri se destaca por adotar o paradigma de *SDN*, empregar uma camada de abstração denominada *Quantum Abstraction Interface (QuAI)* para integrar dispositivos de diferentes fabricantes e permitir a coexistência de canais quânticos e clássicos sobre a mesma infraestrutura [García Cid et al. 2021]. Em conjunto, essas características fazem dela um exemplo relevante de rede metropolitana orientada não apenas à conectividade, mas também à interoperabilidade e à orquestração de serviços.

Além da infraestrutura, a rede de Madri é relevante por já ter sido usada como base para diferentes cenários de aplicação. O próprio estudo relata a implementação ou a preparação de casos de uso associados à NFV baseada em QKD, IPsec e mecanismos de *Ordered Proof of Transit (OPoT)*, o que a aproxima diretamente das classes de uso discutidas no Y Suppl. 80, em especial aquelas relativas à integração com TCP/IP e a formas de rede como SDN/NFV [García Cid et al. 2021, ITU 2023]. Por essa razão, a rede de Madri é um bom exemplo para este capítulo: ela mostra que a QKDN metropolitana não precisa ser entendida apenas como um experimento físico de distribuição de chaves, mas também como uma plataforma de integração com serviços de rede existentes.

Outro exemplo é a *Rio Quantum Network*, que representa uma abordagem metropolitana ainda em construção, com forte ênfase em flexibilidade topológica e na adoção de protocolos que dispensam nós confiáveis na forma tradicional. O projeto conecta instituições de pesquisa no estado do Rio de Janeiro por meio de fibras ópticas da Rede Rio/FAPERJ e de um enlace em espaço livre de 7 km entre o CBPF e a UFF, com base em

uma implementação em laço Sagnac do protocolo TF-MDI-QKD [Temporão et al. 2024]. A rede é reconfigurável, permite que diferentes instituições assumam os papéis de Alice e Bob e concentra em Charlie a maior parte do hardware quântico, o que a torna um caso particularmente interessante para a discussão de redes metropolitanas orientadas a MDI-QKD e de integração híbrida entre fibra óptica e espaço livre [Temporão et al. 2024].

Em outra direção, o *Boston-Area Quantum Network* (BARQNET) configura-se como um *testbed* metropolitano voltado menos à provisão imediata de serviços de chaves e mais à caracterização de enlaces e à integração de componentes para futuras demonstrações de redes quânticas. O sistema conecta o MIT Lincoln Laboratory, o MIT e Harvard por meio de enlaces de fibra comerciais, em uma topologia de três nós, com cerca de 50 km de extensão entre as extremidades e diferentes configurações de operação, incluindo modo diferencial, ida e volta e arranjo de três nós [Bersin et al. 2024]. Seu interesse, no contexto desta discussão, reside em evidenciar que redes metropolitanas também podem atuar como ambientes de validação experimental para aspectos como sincronização, ruído de fase, ruído de polarização e protocolos de distribuição de qubits, complementando, assim, a perspectiva mais diretamente orientada a QKDN observada nos casos de Madri e Rio [Bersin et al. 2024].

Em síntese, as redes metropolitanas mostram por que a QKD precisa ser tratada em nível de arquitetura de rede. Elas exigem integração entre enlaces quânticos e infraestrutura clássica, mecanismos de gerenciamento de chaves, coordenação operacional e, em muitos casos, flexibilidade para acomodar heterogeneidade de dispositivos e múltiplos serviços. Nesse conjunto, a rede de Madri se destaca como referência particularmente útil para este capítulo por combinar escala metropolitana, integração com infraestrutura real de telecomunicações, heterogeneidade tecnológica e alinhamento com casos de uso próximos aos discutidos no ITU-T Y Suppl. 80.

4.5. Comparativo entre ITU-T e ETSI

Nesta subseção, apresenta-se uma análise comparativa entre os *frameworks* normativos desenvolvidos pela ETSI (Seção 4.3.1) e pela ITU-T (Seção 4.4), ambos centrais para a padronização das redes de QKD. Embora compartilhem o propósito de favorecer a interoperabilidade, a segurança e a escalabilidade, essas organizações adotam perspectivas arquiteturais distintas, porém complementares. A ETSI dedica-se principalmente à definição de interfaces e mecanismos de entrega de chaves voltados à integração com aplicações e sistemas clássicos, ao passo que a ITU-T propõe modelos arquiteturais, funcionais e de gerenciamento orientados à operação de redes QKDN em larga escala. Nesse contexto, a comparação entre essas abordagens contribui para compreender como a padronização internacional da QKD vem sendo estruturada de forma coordenada, em camadas e com foco na interoperabilidade.

4.5.1. Perspectivas Arquiteturais Distintas

A principal distinção entre as abordagens reside no *ponto de vista* adotado por cada organismo. A ITU-T, por meio da série Y.3800, enxerga o gerenciamento de chaves como uma **função intrínseca da rede**: a arquitetura em camadas funcionais, quântica, de chaves e de serviços, foi concebida para garantir que a infraestrutura física seja capaz de gerar, ro-

tear e controlar chaves em cenários de múltiplos nós, com controle de recursos e garantias de tráfego quântico.

A ETSI, por outro lado, adota a perspectiva de quem *consome* a infraestrutura. Seu KMS é projetado como uma camada de abstração entre os módulos QKD e as aplicações, oferecendo uma interface padronizada que oculta a complexidade do hardware quântico subjacente: a rede quântica é tratada como um "provedor de chaves", e as aplicações interagem com ela por meio de contratos de serviço bem definidos, seja via chamadas de função (ETSI GS QKD 004) ou via API REST (ETSI GS QKD 014). A distinção central entre os dois *frameworks* é, portanto, de perspectiva a ITU-T define *como* a rede gera e roteia chaves internamente; a ETSI define *como* as aplicações as solicitam e consomem.

4.5.2. Complementaridade dos Modelos

Longe de serem concorrentes, os dois modelos foram concebidos para atuar em camadas distintas da mesma pilha tecnológica, o que os torna **naturalmente complementares**. Uma rede QKD de produção pode, por exemplo, empregar a arquitetura ITU-T para gerenciar toda a lógica interna de seus nós, roteamento de chaves, controle de enlaces quânticos e balanceamento de carga, e, simultaneamente, expor interfaces ETSI nos pontos terminais da rede, permitindo que usuários finais, como roteadores IPsec ou servidores TLS, obtenham suas chaves de forma padronizada e transparente.

Essa complementaridade fica ainda mais evidente quando se considera a relação entre as próprias normas ETSI. A norma ETSI GS QKD 014, que define a API REST de alto nível (discutida na Seção 4.3.3), pode ser implementada *sobre* a norma ETSI GS QKD 004 (Seção 4.3.2): nesse arranjo, a interface REST recebe as requisições das aplicações, as traduz para as chamadas de função da 004 (`OPEN_CONNECT`, `GET_KEY`, etc.) e retorna os resultados no formato JSON esperado pelo cliente. O resultado é uma pilha coerente que unifica gestão de baixo nível e entrega de alto nível em uma única solução integrada, conforme ilustrado na Figura 4.16.

4.5.3. Integração com Protocolos de Segurança Existentes

Um aspecto frequentemente subestimado é a forma como as chaves distribuídas por essa pilha integrada se conectam aos protocolos de segurança já amplamente implementados nas redes convencionais. Protocolos como o **SKIP** (*Simple Key Infrastructure Protocol*) foram projetados exatamente para preencher essa lacuna: atuam como uma camada de adaptação capaz de injetar material criptográfico gerado por QKD em mecanismos de segurança consolidados, como o **IKEv2**, protocolo de troca de chaves utilizado em túneis IPsec [Dervisevic et al. 2025].

Essa integração é relevante porque permite que organizações adotem QKD de forma incremental, sem a necessidade de substituir toda a infraestrutura de segurança existente. O *Key Provider*, seja ele gerenciado conforme a arquitetura ITU-T, a norma ETSI 004 ou a norma ETSI 014, sincroniza o material criptográfico e o disponibiliza aos protocolos tradicionais, que passam a beneficiar-se da segurança incondicional da distribuição quântica de chaves, sem qualquer alteração em sua lógica interna.

4.5.4. Resumo Comparativo

A Tabela 4.1 sintetiza os principais pontos de distinção e complementaridade entre os dois *frameworks*, servindo como referência rápida para os conceitos discutidos ao longo deste capítulo.

Tabela 4.1. Comparação entre os *frameworks* ITU-T e ETSI para redes QKD.

Critério	ITU-T (Série Y.3800)	ETSI (GS QKD 004/014)
Foco principal	Infraestrutura e operação da rede	Entrega de chaves às aplicações
Perspectiva	Provedor de rede	Consumidor / desenvolvedor de aplicações
Modelo de gestão	Função intrínseca da rede (camadas funcionais)	Serviço acessível por API (KMS)
Interface de acesso	Interna à rede	Chamadas de função (004) ou REST/HTTPS (014)
Escalabilidade	Redes de múltiplos nós e roteamento quântico	Alta escalabilidade via APIs web padronizadas
Integração	Controle de recursos e enlaces quânticos	Integração com IPsec, TLS e protocolos legados
Relação mútua	Define *como* as chaves são geradas e roteadas	Define *como* as chaves são consumidas

A análise comparativa revela que a padronização internacional de QKD está sendo construída de forma estratificada e colaborativa. Não há sobreposição de escopo: cada organismo contribui com aquilo que conhece melhor, a ITU-T com sua tradição em arquiteturas de rede de telecomunicações, e a ETSI com sua expertise em interfaces de aplicação e ambientes de desenvolvimento web.

Para o profissional ou pesquisador que deseja implementar uma solução QKD real, a mensagem prática é clara: compreender ambos os *frameworks* não é opcional. A escolha de um deles em detrimento do outro não é uma decisão de arquitetura, mas sim um equívoco conceitual. Uma implantação robusta e interoperável demandará, invariavelmente, elementos de ambos os mundos.

4.6. Segurança do KMS e Avaliação de Confiança

Esta seção analisa os aspectos de segurança relacionados à gestão de chaves em redes QKD, com base nas normas ETSI GS QKD004 [ETSI 2020] e 014 [ETSI 2019] e no protocolo SKIP [Singh et al. 2025]. A discussão está organizada em três eixos complementares: as ameaças inerentes à arquitetura de nós confiáveis, os mecanismos de padronização e certificação de segurança, e a integração do KMS em ambientes híbridos que combinam QKD com criptografia clássica e pós-quântica.

4.6.1. Ameaças Específicas à Gestão de Chaves

As normas analisadas não apresentam uma taxonomia formal de ameaças, mas delineiam um modelo de segurança baseado em **nós confiáveis** e **perímetros controlados**, do qual

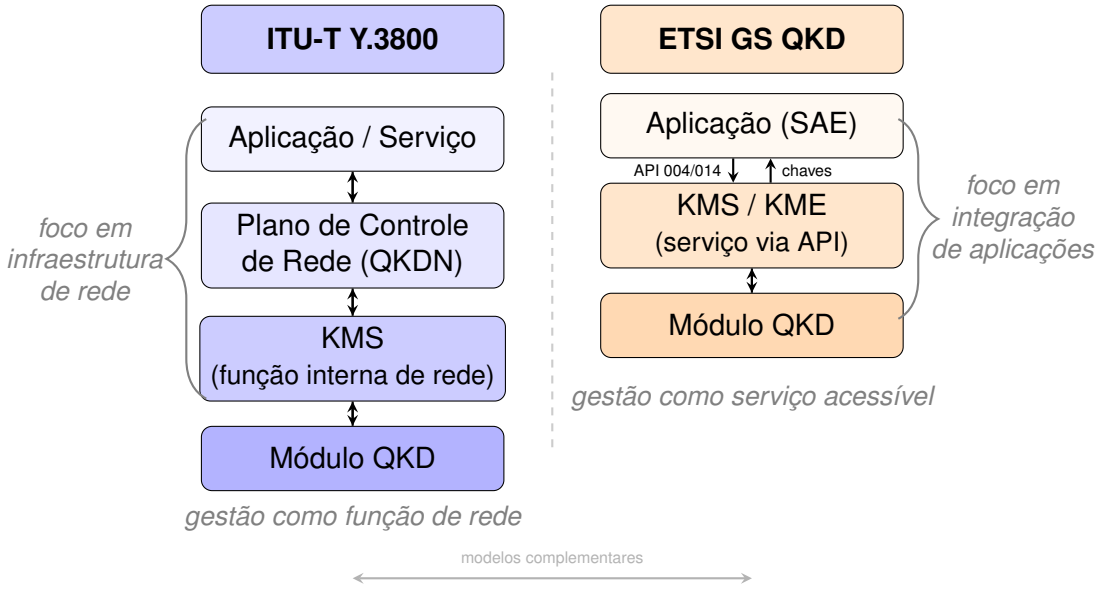


Figura 4.16. Comparação entre os modelos ITU-T e ETSI para gestão de chaves QKD: o ITU-T trata o KMS como função interna de rede, enquanto o ETSI o expõe como serviço acessível por API, tornando os modelos complementares.

decorrem três categorias principais de risco. Compreendê-las é essencial para avaliar a segurança prática de implantações QKD.

Comprometimento de nós confiáveis. A arquitetura ETSI assume que cada Nó Confiável (*Trusted Node*) é um ponto intermediário da rede responsável por armazenar e retransmitir chaves entre enlaces quânticos e é operado e gerenciado de forma segura dentro de um limite fisicamente controlado [ETSI 2020, ETSI 2019]. Essa premissa é, ao mesmo tempo, uma força e uma fragilidade do modelo: se um nó for comprometido, o sigilo das chaves que passaram por ele não pode mais ser garantido. O protocolo SKIP acrescenta uma dimensão adicional a esse risco ao apontar que o uso de identificadores persistentes de sessão pode vazar informações sobre a topologia da rede, facilitando *comprometimentos laterais*, situações em que um adversário, ao observar padrões de tráfego, identifica novos pontos vulneráveis de inserção na rede [Singh et al. 2025].

Segurança do perímetro. Tanto a ETSI GS QKD004 quanto a 014 assumem que a comunicação entre a aplicação consumidora de chaves (SAE) e o gerenciador de chaves (KME/KMS) ocorre dentro de um **perímetro de segurança local**, cuja integridade é pré-requisito para o funcionamento correto do protocolo [ETSI 2020, ETSI 2019]. Em termos práticos, isso significa que, embora a norma exija proteção lógica via HTTPS/TLS, ela pressupõe que essa comunicação ocorra dentro de um perímetro em que controles físicos e de acesso garantam a integridade dos terminais entre a SAE e o KME: essa proteção é responsabilidade da infraestrutura de rede local, tipicamente assegurada por controles físicos e de acesso lógico.

Proteção física e lógica. O SKIP destaca a importância de **co-localizar** o *Key Provider* e o encriptador ou, idealmente, executá-los no mesmo dispositivo físico como medida para reduzir a superfície de ataque no enlace entre esses dois componentes [Singh et al. 2025]. A intuição por trás dessa recomendação é basicamente: quanto

menor o caminho percorrido pela chave entre sua origem e seu uso, menor a janela de oportunidade para um adversário interceptá-la ou adulterá-la.

4.6.2. Padronização de Segurança

A segurança de um sistema QKD não depende apenas de sua implementação correta, mas também de sua capacidade de ser avaliada e certificada por terceiros de forma independente. Nesse sentido, a norma ETSI GS QKD 004 foi projetada com um *footprint* reduzido, isto é, com uma superfície de API deliberadamente pequena e bem delimitada, o que simplifica o escopo de uma auditoria formal de segurança e facilita processos de certificação [ETSI 2020]. Em termos práticos, uma API com menos funções e estados possíveis é mais fácil de verificar formalmente e menos propensa a vulnerabilidades introduzidas por complexidade desnecessária.

Complementarmente, o anexo bibliográfico da norma 004 refere-se ao *National Institute of Standards and Technology (NIST) Special Publications* 800-53 [ETSI 2020], um catálogo amplamente adotado de controles de segurança e privacidade para sistemas de informação, desenvolvido pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos. Essa referência sinaliza a intenção de alinhamento com *frameworks* de conformidade já estabelecidos, facilitando a adoção de QKD em organizações que já operam sob esses regimes regulatórios.

4.6.3. KMS em um Contexto Híbrido (QKD + Criptografia Clássica)

Este é, sem dúvida, um dos aspectos mais estratégicos da gestão de chaves em redes QKD, e também o mais detalhado nas fontes analisadas especialmente no protocolo SKIP [Singh et al. 2025]. A questão central é: como integrar a segurança incondicional fornecida pelo QKD à infraestrutura criptográfica clássica já amplamente implantada? A resposta é apresentada em três camadas complementares.

QKD como fonte de entropia (QRNG). O protocolo SKIP define explicitamente um método denominado `GetEntropy`, que permite ao *Key Provider* fornecer uma cadeia de bits verdadeiramente aleatórios gerados por processos quânticos para consumo interno dos encriptadores [Singh et al. 2025]. O encriptador pode solicitar tamanhos específicos de entropia via API, tornando o hardware quântico uma fonte de aleatoriedade certificável para sistemas clássicos. Isso é relevante porque a qualidade da aleatoriedade é um dos fatores mais críticos para a segurança de qualquer sistema criptográfico: geradores pseudoaleatórios clássicos podem apresentar padrões exploráveis por adversários suficientemente poderosos.

KMS como ponto de convergência com a Criptografia Pós-Quântica (PQC). O SKIP é apresentado como um protocolo de integração: o *Key Provider* pode utilizar tanto QKD quanto algoritmos de criptografia pós-quântica como os padronizados recentemente pelo NIST, como CRYSTALS-Kyber como fonte de material criptográfico, de forma completamente transparente para o encriptador [Singh et al. 2025]. Essa característica posiciona o KMS como um ponto de convergência natural entre os dois paradigmas de segurança quântica, permitindo que organizações adotem gradualmente as tecnologias mais adequadas ao seu contexto sem alterar a lógica dos sistemas que as consomem.

Defesa em profundidade. A abordagem mais poderosa descrita nas fontes

combina QKD e criptografia clássica em uma estratégia de **defesa em profundidade** (*defense-in-depth*): chaves fornecidas pelo KMS são injetadas no processo de derivação de chaves de protocolos consolidados, como IKEv2 (usado em túneis IPsec) e TLS [Singh et al. 2025, ETSI 2020]. Essas chaves pré-compartilhadas são resistentes ao algoritmo de Grover[5], reduzindo efetivamente pela metade o comprimento de bit da segurança de cifras simétricas. Uma chave AES-256, por exemplo, oferece segurança equivalente a 128 bits contra um adversário com computador quântico.O principal algoritmo quântico que ameaça cifras simétricas é integrado ao material de chaveamento já existente, sem exigir a substituição imediata de toda a infraestrutura. O resultado é um sistema que permanece seguro mesmo diante de adversários equipados com computadores quânticos, ao mesmo tempo em que preserva a compatibilidade com os protocolos e os equipamentos já implantados.

Em síntese, os três eixos discutidos nesta seção revelam uma visão de segurança madura e estratificada: as normas reconhecem os limites físicos e arquiteturais do modelo de nós confiáveis, buscam facilitar a certificação formal por meio de APIs enxutas e posicionam o KMS não como substituto da criptografia clássica, mas como uma camada adicional de segurança que fortalece e complementa a infraestrutura existente.

4.7. Laboratório prático em GNS3 para criação de tunel IPsec, conforme ETSI GS QKD 014

Este módulo apresenta a implementação prática de uma infraestrutura de rede *Quantum-Safe* em ambiente de simulação, focando na integração entre um KME e sistemas de rede clássicos para o estabelecimento de túneis seguros. O objetivo final é alcançar um laboratório com uma configuração como a mostrada na figura 4.17.

4.7.1. Visão Geral e Tecnologias Utilizadas

O laboratório simula um cenário onde a segurança de uma VPN IPsec é reforçada por chaves quânticas distribuídas via API REST. As principais tecnologias empregadas incluem:

- **GNS3:** Plataforma de simulação para a orquestração da topologia de rede.
- **MikroTik RouterOS (v7.22+):** Atua como o elemento de rede (*Secure Application Entity* - SAE), possuindo suporte nativo ao protocolo QKD para consumo de chaves externas.
- **Docker:** Utilizado para instanciar a KME de forma isolada e portável.
- **Python/Flask:** Linguagem e *framework* utilizados para desenvolver a lógica da KME, implementando a norma **ETSI GS QKD 014** para a entrega de chaves.
- **SQLite:** Base de dados leve para a persistência e sincronização do material criptográfico gerado.

[5]O algoritmo de Grover, proposto em 1996, é um algoritmo quântico capaz de realizar buscas em espaços não estruturados com complexidade $O(\sqrt{N})$.

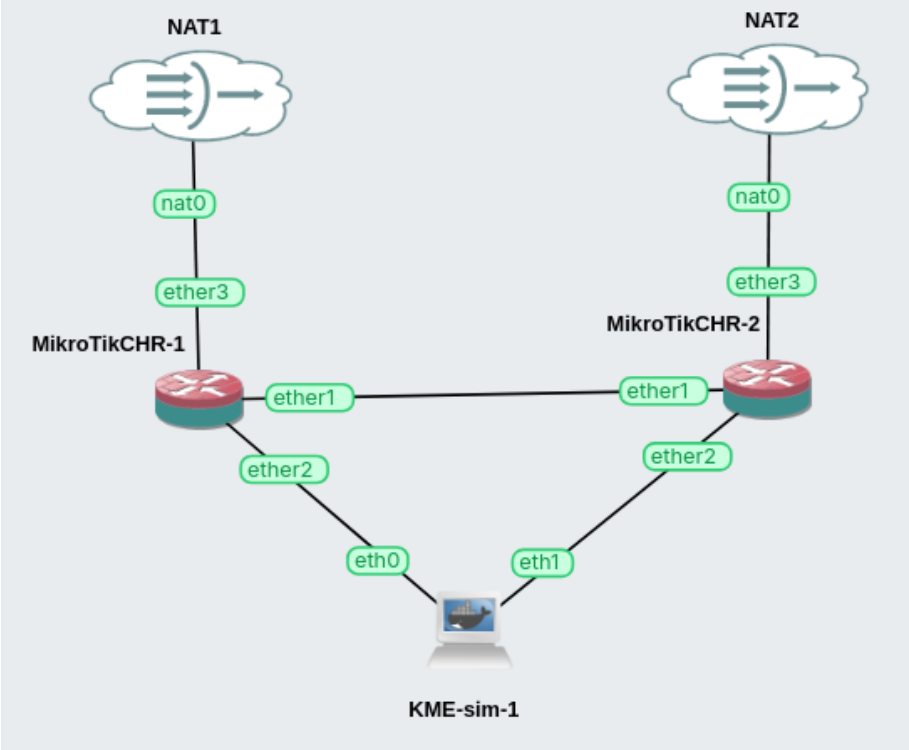


Figura 4.17. Imagem da simulação final no GNS3.

4.7.2. Topologia de Rede

A topologia é composta por três núcleos principais interconectados em um ambiente isolado:

- **Backbone Clássico:** Dois roteadores MikroTik (CHR-1 e CHR-2) conectados por um enlace ponto-a-ponto (sub-rede `10.0.0.0/30`).
- **Rede de Gerenciamento (KME-LAN):** Uma infraestrutura centralizada onde o servidor KME reside, acessível por ambos os roteadores através de interfaces dedicadas em suas respectivas LANs (`172.16.1.0/24` e `172.16.2.0/24`).
- **Plano de Dados:** Onde o tráfego do usuário é cifrado via IPsec utilizando a entropia fornecida pela KME.

4.7.3. Decisões Arquiteturais

O *design* do laboratório reflete escolhas estratégicas para garantir a segurança e a conformidade com padrões internacionais:

1. **Centralização da KME:** Diferente de modelos puramente descentralizados, utiliza-se uma KME central com interfaces múltiplas para simplificar a sincronização de chaves em ambiente de teste, atuando como o *middleware* entre a geração de entropia e o consumo.

2. **Autenticação via mTLS:** O acesso à API de chaves é protegido por TLS Mútuo (mTLS), garantindo que apenas roteadores autorizados (SAEs) possam solicitar material criptográfico.

3. **Separação de Planos:** O plano de gerenciamento de chaves (REST/HTTPS) é logicamente separado do plano de dados (IPsec), seguindo as recomendações de demarcação de segurança da série ITU-T Y.3800.

4. **Consumo Híbrido:** O sistema é configurado para que o IPsec utilize as chaves da KME como material primário, provendo o *Forward Secrecy* contra ameaças quânticas (ataques do tipo *Harvest Now, Decrypt Later*).

4.7.4. Fluxo de Operação

O funcionamento do laboratório segue o ciclo de vida da chave: a KME gera e armazena chaves sob demanda no SQLite; os roteadores MikroTik, agindo como clientes mTLS, solicitam essas chaves via `GET requests` na porta `8020`; por fim, o subsistema IPsec do RouterOS utiliza o `key_id` recebido para sincronizar a cifragem do túnel entre as duas pontas.

4.7.5. Artefatos

A implementação do laboratório, assim como um tutorial detalhado pode ser encontrado no seguinte repositório: `https://github.com/QuIIN-Quantum-Industrial-Innovation/lab_mc_sbrc_26`.

4.8. Encerramento, Desafios e Caminhos Futuros:

A jornada percorrida neste texto demonstrou que a segurança *Quantum-Safe* não é apenas uma aspiração teórica, mas uma realidade tecnológica em plena fase de padronização e implementação prática. A transição da criptografia clássica para modelos resilientes à computação quântica exige uma compreensão profunda da integração entre as leis da física e a engenharia de redes.

4.8.1. Síntese dos Conceitos Fundamentais

Um dos componentes centrais nessa arquitetura é o **KMS (Key Management System)**, que atua como o elemento de software vital para superar as limitações da camada física do QKD, restrita por distâncias e topologias ponto-a-ponto. O KMS transforma o fluxo bruto de material criptográfico em um recurso gerenciável, permitindo o armazenamento, o roteamento e a entrega eficiente de chaves para as aplicações finais.

Essa operacionalização é sustentada por uma dualidade de padronização complementar: enquanto a abordagem **ETSI ISG QKD** foca na *interface de serviço*, definindo como as aplicações consomem chaves via APIs (como a REST API na GS QKD 014) para garantir a interoperabilidade entre encriptadores e gerenciadores, a abordagem **ITU-T Série Y.3800** concentra-se na *arquitetura de rede*. Esta última define a interação entre os planos quântico, de chave, de controle e de gerenciamento para formar uma **QKDN** (*Quantum Key Distribution Network*) escalável e de múltiplos saltos.

O sucesso desse modelo reside no desacoplamento funcional, que separa a geração da chave na camada quântica de seu gerenciamento na camada de chaves, permitindo que a infraestrutura de segurança evolua independentemente do hardware. Além disso, a síntese prática demonstra que o QKD não substitui a criptografia clássica, mas a fortalece através de uma estratégia de defesa em profundidade. A integração com protocolos consolidados, como o **IPsec**, ilustra esse modelo de segurança híbrida, onde a segurança teórica da informação provida pelo QKD protege a negociação de chaves em canais clássicos. Por fim, a viabilidade técnica foi validada através de simulações no **GNS3**, comprovando que o uso de normas abertas já permite a construção de redes seguras com ferramentas de mercado, reduzindo as barreiras para administradores de redes tradicionais.

4.8.2. Desafios Tecnológicos e Operacionais

Apesar dos avanços, a implementação em larga escala enfrenta desafios significativos, especialmente no que tange à escalabilidade e à carga administrativa. A instalação manual de chaves pré-compartilhadas (PSKs) apresenta um gargalo, visto que o esforço administrativo cresce de forma quadrática com o aumento de nós, exigindo a transição para métodos de provisionamento dinâmico. Enquanto o modelo ETSI foca na entrega, o modelo **Y.38XX** destaca a complexidade de gerenciar o ciclo de vida das chaves em redes *multi-hop*, o que demanda automação robusta.

Somam-se a isso os obstáculos de interoperabilidade entre equipamentos de diferentes fabricantes, motivando a busca por APIs padronizadas. No nível físico e logístico, a segurança do "último salto" entre o Provedor de Chaves (KP) e o encriptador permanece crítica, recomendando-se a co-localização física ou a integração em um mesmo dispositivo para mitigar ataques. Há também a dependência de perímetros de segurança locais para as interfaces de aplicação, um desafio constante em redes distribuídas. Por fim, os desenvolvedores enfrentam um *trade-off* entre complexidade e analisabilidade: a norma ETSI 014 oferece simplicidade via integração *web*, mas exige bibliotecas HTTPS/JSON complexas, enquanto a ETSI 004, embora mais árdua, permite implementação em C puro, facilitando auditorias e certificações de segurança.

4.8.3. Caminhos Futuros e Tendências

O futuro da segurança quântica aponta para a agilidade criptográfica e a modularidade, permitindo que Provedores de Chaves sejam atualizados ou substituídos sem a necessidade de reconfigurar os encriptadores. A tendência de convergência sugere o uso de chaves quânticas para reforçar protocolos como IKEv2 e TLS, combinando a robustez do QKD com a flexibilidade da criptografia pós-quântica (PQC) em uma estratégia híbrida.

A expansão do uso dessas chaves em protocolos de mercado, como **MACsec** e **IPsec**, será facilitada por protocolos de integração como o **SKIP**, automatizando o ciclo de vida do material criptográfico. Paralelamente, o desenvolvimento de APIs REST leves visa democratizar o acesso à tecnologia, permitindo que o QKD seja consumido como um serviço de nuvem (*as-a-service*). Em arquiteturas mais complexas, a integração de servidores de chaves com controladores de Redes Definidas por Software (**SDN**) será essencial para otimizar a descoberta de aplicações e o roteamento de chaves. Por fim, observa-se uma tendência de integração com sistemas legados, onde o QKD passa a servir

como fonte adicional de entropia para sistemas de gerenciamento convencionais, como o padrão **KMIP da OASIS**.

Referências

[ITU 2023] (2023). Supplement 80 to itu-t y-series recommendations: Itu-t y.3800 series – quantum key distribution networks use cases. Technical report, ITU-T.

[Ahmed et al. 2025] Ahmed, N., Zhang, L., and Gangopadhyay, A. (2025). A survey of post-quantum cryptography support in cryptographic libraries. In *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pages 906–917. IEEE.

[Bennett and Brassard 2014] Bennett, C. H. and Brassard, G. (2014). Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11. Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84.

[Bersin et al. 2024] Bersin, E., Grein, M., Sutula, M., Murphy, R., Huan, Y. Q., Stevens, M., Suleymanzade, A., Lee, C., Riedinger, R., Starling, D. J., et al. (2024). Development of a boston-area 50-km fiber quantum network testbed. *Physical Review Applied*, 21(1):014024.

[Bhatia and Ramkumar 2020] Bhatia, V. and Ramkumar, K. R. (2020). An efficient quantum computing technique for cracking rsa using shor's algorithm. In *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, pages 89–94.

[Cao et al. 2022] Cao, Y., Zhao, Y., Wang, Q., Zhang, J., Ng, S. X., and Hanzo, L. (2022). The evolution of quantum key distribution networks: On the road to the qinternet. *IEEE Communications Surveys & Tutorials*, 24(2):839–894.

[Day and Zimmermann 1984] Day, J. and Zimmermann, H. (1984). The osi reference model. *Proceedings of the IEEE*, 71:1334 – 1340.

[Dervisevic et al. 2025] Dervisevic, E., Tankovic, A., Fazel, E., Kompella, R., Fazio, P., Voznak, M., and Mehic, M. (2025). Quantum key distribution networks - key management: A survey. *ACM Comput. Surv.*, 57(10).

[Easttom 2022] Easttom, C. (2022). *Quantum Computing and Cryptography*, pages 397–407. Springer International Publishing, Cham.

[Elboukhari et al. 2010] Elboukhari, M., Azizi, M., and Azizi, A. (2010). Quantum key distribution protocols: A survey. *International Journal of Universal Computer Sciences*, 1(2):59–67.

[ETSI 2019] ETSI (2019). Quantum key distribution (qkd); protocol and data format of rest-based key delivery api. Technical Specification TS 102 014, ETSI. Version 1.1.1, Released: February 2019.

[ETSI 2020] ETSI (2020). Quantum key distribution (qkd); application interface. Technical Specification TS 102 004, ETSI. Version 2.1.1 , Released: August 2020.

[García Cid et al. 2021] García Cid, M. I., Ortiz Martín, L., and Martín Ayuso, V. (2021). Madrid quantum network: A first step to quantum internet. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–7.

[Gisin et al. 2002] Gisin, N., Ribordy, G., Tittel, W., and Zbinden, H. (2002). Quantum cryptography. *Reviews of Modern Physics*, 74(1):145–195.

[ITU-T 2019] ITU-T (2019). Overview on Quantum Key Distribution Networks. Technical Report Recommendation ITU-T Y.3800, International Telecommunication Union. Accessed: 2026-03-08.

[ITU-T 2020a] ITU-T (2020a). Functional Requirements for Quantum Key Distribution Networks. Technical Report Recommendation ITU-T Y.3801, International Telecommunication Union. Accessed: 2026-03-08.

[ITU-T 2020b] ITU-T (2020b). Quantum Key Distribution Networks – Control and Management. Technical Report Recommendation ITU-T Y.3804, International Telecommunication Union. Accessed: 2026-03-08.

[ITU-T 2020c] ITU-T (2020c). Quantum Key Distribution Networks – Functional Architecture. Technical Report Recommendation ITU-T Y.3802, International Telecommunication Union. Accessed: 2026-03-08.

[ITU-T 2020d] ITU-T (2020d). Quantum Key Distribution Networks – Key Management. Technical Report Recommendation ITU-T Y.3803, International Telecommunication Union. Accessed: 2026-03-08.

[James et al. 2023] James, P., Laschet, S., Ramacher, S., and Torresetti, L. (2023). Key management systems for large-scale quantum key distribution networks. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, pages 1–9.

[Lai et al. 2023] Lai, J., Yao, F., Wang, J., Zhang, M., Li, F., Zhao, W., and Zhang, H. (2023). Application and development of qkd-based quantum secure communication. *Entropy*, 25(4):1–18.

[Mehic et al. 2020] Mehic, M., Niemiec, M., Rass, S., Ma, J., Peev, M., Aguado, A., Martin, V., Schauer, S., Poppe, A., Pacher, C., and Voznak, M. (2020). Quantum key distribution: A networking perspective. *ACM Computing Surveys (CSUR)*, 53(5):1–41.

[Narroway 2025] Narroway, e. a. (2025). Towards Global Quantum Key Distribution. *Nature Reviews Physics*.

[Nielsen and Chuang 2010] Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed edition.

[Sáez et al. 2024] Sáez, J. M., Perales, A. P., Palancar, R. C., Lopez, D. R., Chavarria, J. F., Ayuso, V. M., and Mendez, J. P. B. (2024). Current status, gaps, and future directions in quantum key distribution standards: Implications for industry. In *2024 international conference on Quantum Communications, Networking, and Computing (QCNC)*, pages 341–345. IEEE.

[Sanz et al. 2025] Sanz, A., Atutxa, A., Franco, D., Astorga, J., Jacob, E., and López, D. (2025). Toward quantum-safe scalable networks: an open, standards-aware key management framework. *IEEE Network*.

[Singh et al. 2025] Singh, R., Hill, C., Kawaguchi, S., and Lupo, J. (2025). Secure Key Integration Protocol (SKIP). Internet-Draft draft-cisco-skip-02, Internet Engineering Task Force. Work in Progress.

[Stallings 2017] Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, 7th edition.

[Temporão et al. 2024] Temporão, G. P., de Melo, F. R. B., and Khoury, A. Z. (2024). The rio quantum network: a reconfigurable hybrid multi-user metropolitan quantum key distribution network. In *Workshop de Redes Quânticas*, pages 19–24. SBC.

[Xu et al. 2020] Xu, F., Ma, X., Zhang, Q., Lo, H.-K., and Pan, J.-W. (2020). Secure quantum key distribution with realistic devices. *Reviews of modern physics*, 92(2):025002.

Capítulo

5

Construção de Sistemas de Gêmeos Digitais: Uma Abordagem baseada em Middleware

André Almeida (IFRN), Lucas Pereira (IFRN & UFRN), Thais Batista (UFRN), Everton Cavalcante (UFRN), Flavia C. Delicato (UFF), Rebeca Motta (UFF)

Resumo

Os gêmeos digitais (do inglês, digital twins – DTs) são réplicas digitais dinâmicas de objetos físicos, sistemas ou processos do mundo real que são continuamente atualizadas de forma bidirecional com dados de sensores e dispositivos computacionais. Ao refletir, em tempo quase real, o estado e o comportamento de suas contrapartes físicas, DTs possibilitam monitorar, analisar e controlar sistemas complexos. A construção de sistemas de DTs apresenta desafios significativos decorrentes da heterogeneidade das tecnologias e dos dados envolvidos, bem como da complexidade de manter sincronizadas as interações entre elementos físicos e virtuais. Este capítulo tem por objetivo introduzir os principais conceitos e desafios relacionados à construção de sistemas de DTs e discutir como plataformas de middleware podem ser utilizadas nesse contexto. Para ilustrar esse uso na prática, utilizamos o MidDiTS, um middleware projetado especificamente para dar suporte à construção e à operação de sistemas de DTs por meio de um processo sistemático de desenvolvimento.

Abstract

Digital twins (DTs) are dynamic digital replicas of real-world physical objects, systems, or processes that are continually updated bidirectionally with data from sensors and computing devices. By reflecting the state and behavior of their physical counterparts in near real-time, DTs enable the monitoring, analysis, and control of complex systems. Building DT systems presents significant challenges stemming from the heterogeneity of the technologies and data involved, as well as the complexity of synchronizing interactions between physical and virtual elements. This chapter introduces the main concepts and challenges of building DT systems and discusses how middleware platforms can be used in this context. To illustrate this use in practice, we employ MidDiTS, a middleware system specifically designed to support the construction and operation of DT systems through a systematic development process.

5.1. Introdução

A crescente digitalização de sistemas físicos, impulsionada pela proliferação de dispositivos conectados e pela expansão da capacidade computacional, tem dado origem a novas formas de monitorar, compreender e otimizar o comportamento de entidades do mundo real. Nesse cenário, o conceito de *gêmeo digital* (do inglês, *digital twin* – DT) emergiu como uma das tecnologias mais promissoras da Indústria 4.0 e da transformação digital contemporânea, convergindo paradigmas e tecnologias como a Internet das Coisas (IoT), sistemas ciberfísicos, computação em nuvem, simulação computacional e Inteligência Artificial para viabilizar uma integração dinâmica entre os domínios físico e digital.

A Norma Internacional ISO/IEC 30173:2023 formaliza a definição de um DT como a "representação digital de uma entidade-alvo com conexões de dados que permitem a convergência entre estados físicos e digitais a uma taxa apropriada de sincronização" [ISO 2023]. Sob essa ótica, um DT não é apenas um modelo estático, mas uma réplica digital dinâmica, baseada em software, continuamente atualizada com dados provenientes de sensores e dispositivos computacionais ao longo de seu ciclo de vida. Essa integração permite que o modelo virtual reflita, em tempo quase real, o estado e o comportamento de seu correspondente no mundo real. No entanto, um diferencial crítico de um DT em relação aos modelos tradicionais de simulação computacional ou aos modelos digitais construídos apenas a partir dos dados coletados das entidades do mundo real (ao que a literatura denomina sombras digitais) é a sincronização bidirecional automatizada de dados entre os mundos real e virtual [Kritzinger et al. 2018]. Dessa forma, um DT propriamente dito permite que as mudanças na entidade do mundo real sejam refletidas na representação virtual e vice-versa, possibilitando não apenas o monitoramento, mas também o controle e a otimização direta da entidade-alvo.

Estruturalmente, um sistema de DT constitui-se como um ecossistema computacional distribuído que integra modelos digitais, mecanismos de comunicação, serviços de sincronização e componentes de análise e de atuação [Liu et al. 2021]. Essas funcionalidades têm motivado a adoção de sistemas de DTs em diversos domínios de aplicação [Fuller et al. 2020, Iranshahi et al. 2025]. Por exemplo, na indústria de manufatura, sistemas de DTs são utilizados para monitoramento em tempo real e manutenção preditiva de ativos, bem como para otimização de processos produtivos [Tao et al. 2019]. Em cidades inteligentes, sistemas de DTs permitem simular cenários urbanos complexos a partir de dados coletados por sensores de IoT, contribuindo para um planejamento e uma gestão mais eficientes [Shahat et al. 2021]. No setor agrícola, sistemas de DTs viabilizam a criação de réplicas digitais de propriedades rurais inteiras, integrando dados de sensores de solo, condições de plantas, previsões meteorológicas e imagens de satélite para apoiar decisões sobre irrigação, fertilização e manejo de pragas [Pylianidis et al. 2021].

Apesar do seu potencial, o desenvolvimento de sistemas de DTs envolve um processo complexo com múltiplos desafios. Do ponto de vista técnico, destacam-se a necessidade de modelagem precisa das entidades físicas, a integração de fontes de dados heterogêneas, a garantia de comunicação contínua entre múltiplas plataformas e a manutenção da sincronização em tempo real entre os mundos físico e virtual. Esses desafios são ampliados pela diversidade de domínios de aplicação, cada qual com requisitos próprios de latência, escala, segurança e interoperabilidade.

Soluções de *middleware* desempenham um papel importante nesse contexto justamente para endereçar alguns desses desafios. Originalmente concebido para simplificar o desenvolvimento de sistemas distribuídos, um *middleware* atua como uma camada de abstração que fornece interfaces padronizadas e serviços em tempo de execução, abstraindo as heterogeneidades de comunicação e de distribuição [Coulouris et al. 2011]. No contexto de sistemas de DTs, essa função torna-se ainda mais relevante, pois tais sistemas precisam integrar componentes de natureza, origem e protocolo distintos, incluindo sensores físicos, modelos de simulação, serviços de análise e interfaces de atuação. Portanto, um *middleware* promove a interoperabilidade entre esses componentes e reduz significativamente a complexidade de desenvolvimento. No entanto, é crucial que seu uso seja guiado por um processo de desenvolvimento claro e sistemático, que oriente os desenvolvedores ao longo das etapas de concepção e implementação de sistemas de DTs.

Nesse contexto, este capítulo tem por objetivo introduzir os principais conceitos e desafios relacionados a sistemas de DTs e demonstrar o uso de um *middleware* para a construção desses sistemas. Para tanto, utilizaremos o MidDiTS [Pereira et al. 2024], um *middleware* projetado especificamente para o desenvolvimento de sistemas de DT. O MidDiTS visa atender a características essenciais desses sistemas, tais como a modelagem de entidades físicas e seus atributos, a manutenção da sincronização contínua entre estados físicos e digitais, a integração com múltiplas fontes de dados e plataformas heterogêneas, a disponibilização de interfaces padronizadas para monitoramento e atuação e o suporte à análise e à evolução dos DTs ao longo do ciclo de vida da entidade-alvo. Essas características estão alinhadas aos requisitos estabelecidos pelo *Digital Twin Consortium* (DTC)[1], um consórcio internacional que reúne dezenas de centros de pesquisa e empresas de diferentes áreas, voltado à definição de diretrizes e boas práticas para sistemas de gêmeos digitais (DTs).

O restante deste capítulo está organizado da seguinte forma. A Seção 5.2 discute os requisitos arquiteturais e os desafios de implementação de sistemas de DTs. A Seção 5.3 descreve como soluções baseadas em *middleware* podem dar suporte ao desenvolvimento de sistemas de DTs. A Seção 5.4 apresenta uma visão geral da arquitetura do *middleware* MidDiTS e de sua implementação. A Seção 5.5 descreve as etapas do processo de construção de sistemas de DTs com o MidDiTS. Por fim, a Seção 5.6 apresenta algumas considerações finais.

5.2. Desafios e Requisitos de Sistemas de Gêmeos Digitais

Embora o conceito de DTs tenha um potencial transformador, o desenvolvimento dessas soluções ainda é um processo complexo que enfrenta desafios significativos. Parte dessa complexidade está associada ao projeto e à implementação dos elementos que compõem essas soluções [Pereira et al. 2024]. Entre esses elementos, destacam-se as *entidades de DT*, que se referem a ativos físicos, processos ou ambientes no mundo real (por exemplo, um sensor, uma sala ou uma casa inteligente) para os quais é necessária uma representação virtual. Para compor a solução, define-se uma *instância de DT* como a representação digital que se mantém ativamente sincronizada com sua contraparte física. Por fim, um *sistema de DT* é a aplicação ou ecossistema abrangente que agrega e orquestra múlti-

[1] https://www.digitaltwinconsortium.org

plas instâncias de DT interconectadas e funcionalidades voltadas para oferecer suporte a necessidades específicas [van Schalkwyk et al. 2025].

A distinção entre esses elementos é relevante para separar o elemento do mundo real (entidade de DT) de sua contraparte digital operacional (instância de DT). Assim, com a necessidade de manter o sincronismo entre os elementos de forma contínua e precisa, diferentes desafios se apresentam [Almeida et al. 2023]:

- **Modelagem.** A literatura destaca a necessidade de melhorar metodologias e técnicas para o desenvolvimento de sistemas de DTs. No entanto, a falta de uma compreensão mais profunda sobre a conceituação e a modelagem de DTs dificulta a plena realização deste aspecto.
- **Interoperabilidade.** Sistemas de DTs lidam constantemente com dados provenientes de fontes altamente heterogêneas, frequentemente produzidos por diferentes fabricantes e utilizando protocolos diversos. A integração contínua dessas tecnologias e aplicações representa um desafio de interoperabilidade para essas soluções.
- **Sincronização.** Manter interações sincronizadas e preservar a comunicação bidirecional em tempo real entre as entidades do mundo real e as suas respectivas representações virtuais exige o processamento rápido de eventos e garantias de baixa latência nas redes de comunicação.
- **Orquestração.** A comunicação transparente entre diferentes elementos em um sistema de DT, bem como a coordenação dos fluxos de dados e das interações entre múltiplas entidades do mundo real e instâncias virtuais, são desafios que afetam a maturidade e a adoção do paradigma.

Para mitigar esses desafios e guiar o desenvolvimento a partir de necessidades de uso, o DTC organizou os requisitos e capacidades fundamentais de um ecossistema de DTs em seis categorias principais [van Schalkwyk et al. 2025]:

1) **Serviços de Dados.** Requisitos voltados para viabilizar o acesso, a ingestão e o gerenciamento de dados, da borda até a nuvem. O desafio da representação e da modelagem de dados pode ser tratado dentro desta categoria, indicando a necessidade de abstração de modelagem adaptada a diferentes domínios. Do ponto de vista da infraestrutura, esses serviços incluem armazenamento, aquisição e processamento de dados em tempo real ou em lotes. O gerenciamento de dados específicos de domínio, incluindo dados de séries temporais e dados geoespaciais, insere-se também nesta categoria.
2) **Integração.** Em resposta ao desafio da interoperabilidade, esta categoria preocupa-se em habilitar o acesso a dados e a comunicação com sistemas e aplicações pré-existentes, tanto internas quanto externas. Os requisitos desta categoria englobam a integração de sistemas, a interoperabilidade e a integração direta entre diferentes sistemas de DTs.
3) **Inteligência.** Um dos principais usos dos DTs é replicar processos do mundo real por meio de simulações. Técnicas de análise de dados e inteligência computacional podem enriquecer a representação virtual desses processos. Assim, esta

categoria considera o ambiente e as ferramentas para o desenvolvimento e a implantação de soluções de DTs, incluindo orquestração de fluxos, execução de simulações, modelagem de predição, análise de dados e integração com Inteligência Artificial.

4) **Experiência de Usuário.** Esta categoria engloba requisitos que garantem às pessoas usuárias a capacidade de interagir com os DTs e visualizar seus dados de maneira eficiente. Isso inclui a visualização das entidades e o monitoramento em tempo real com *Business Intelligence*, abarcando desde painéis de controle (*dashboards*) até interfaces avançadas, como a realidade aumentada.
5) **Gerenciamento.** Esta categoria compreende as capacidades voltadas para a governança e a manutenção do ecossistema de DTs. A capacidade de monitorar continuamente o ambiente e seus elementos é crucial para garantir o bom funcionamento de um DT, e os requisitos operacionais incluem o gerenciamento contínuo de dispositivos, o monitoramento abrangente do sistema, o registro de eventos e políticas de governança de dados. A orquestração é outra funcionalidade relacionada ao gerenciamento e engloba a coordenação das interações e dos fluxos de dados entre os diversos elementos de um sistema de DT, os quais podem variar de ativos físicos a aplicações corporativas.
6) **Confiabilidade.** Capacidades de segurança, privacidade e criptografia de dados são aspectos essenciais para DTs, principalmente nos domínios de aplicação em que são utilizados. A confiabilidade abrange requisitos críticos que asseguram a adoção segura e responsável da tecnologia, englobando desde segurança corporativa, segurança nativa dos dispositivos e criptografia de dados, até privacidade, resiliência e proteção, incluindo especificamente questões de autenticação, autorização e controle de acesso.

Dentro dessa iniciativa, o DTC também propôs a Tabela Periódica de Capacidades de Gêmeos Digitais [van Schalkwyk et al. 2025]. Essa tabela atua como um *framework* agnóstico de arquitetura e tecnologia para a definição de requisitos para DTs. Nela, as seis categorias anteriormente mencionadas são expandidas em um conjunto de 62 requisitos de alto nível e capacidades técnicas necessárias para compor soluções complexas de DTs.

Apesar da abrangência da proposta do DTC, algumas considerações sobre as relações entre os requisitos merecem atenção. Por exemplo, a sincronização entre uma instância de DT e seu ativo físico (isto é, a manutenção da correspondência bidirecional e em tempo real entre o mundo físico e sua representação digital) é um requisito que atravessa as categorias de Integração e Gerenciamento. Do ponto de vista da conectividade e do transporte de dados, a sincronização é um problema de Integração; já do ponto de vista da consistência de estado e do monitoramento da qualidade dessa correspondência ao longo do tempo, ela se aproxima do Gerenciamento. Outro aspecto a ser considerado diz respeito às capacidades de orquestração e alocação de recursos, que envolvem decisões sobre onde implantar instâncias de DT e como alocar recursos computacionais para sua execução. A orquestração pode ser associada à categoria de Inteligência, mas os mecanismos de orquestração pertencem ao Gerenciamento, uma vez que diversas estratégias de orquestração operam como políticas de infraestrutura e governança do ecossistema.

As seis categorias de requisitos anteriormente apresentadas podem orientar a avali-

ação e o desenvolvimento de soluções para sistemas de DTs. Contudo, identificar como esses requisitos podem ser concretamente atendidos na prática ainda representa um grande desafio. Diante desse amplo espectro de capacidades, é necessária uma abordagem em que capacidades técnicas individuais das entidades possam ser modeladas em DTs específicos e orquestradas como um ecossistema de DTs. Nesse sentido, plataformas de *middleware* emergem como uma camada de abstração estratégica, capaz de encapsular e prover parte dessas capacidades de forma padronizada e reutilizável, sendo esse o foco das próximas seções deste capítulo.

5.3. Uso de Middleware em Sistemas de Gêmeos Digitais

A evolução das plataformas de *middleware* está fortemente associada ao surgimento de sistemas distribuídos e à necessidade crescente de abstrair sua inerente complexidade. Desde os anos 1990, com o avanço das redes de computadores, tornou-se evidente a demanda por camadas intermediárias de software capazes de facilitar a comunicação, a coordenação e a integração entre componentes heterogêneos, muitas vezes executando em diferentes plataformas e localizações [Bernstein 1996]. Nesse contexto, o papel primordial de um *middleware* é preencher a lacuna funcional entre os programas de aplicação e a complexa infraestrutura de comunicação e distribuição subjacente.

Um *middleware* atua para coordenar a conexão e a interoperabilidade entre as partes de uma aplicação, além de simplificar a integração de componentes distribuídos desenvolvidos por múltiplos provedores [Schantz and Schmidt 2002]. Ao abstrair a heterogeneidade e proteger os desenvolvedores de detalhes complexos de baixo nível, essa camada torna viável e eficiente o desenvolvimento e a evolução de sistemas distribuídos robustos. Em essência, o *middleware* é projetado para isolar os desenvolvedores das complexidades inerentes à distribuição e à heterogeneidade, que são fontes recorrentes de erros em sistemas distribuídos.

Apesar da aplicabilidade e eficácia das tecnologias de *middleware* de propósito geral, certos domínios exigem serviços específicos não tratados pelas plataformas convencionais. Essas plataformas são tipicamente genéricas e, portanto, não endereçam nativamente os requisitos semânticos complexos de novos paradigmas, principalmente no contexto de aplicações industriais. Para cobrir essa lacuna, na última década foram propostas soluções de *middleware* voltadas para domínios específicos, como a IoT, incorporando serviços para lidar com o novo contexto de integração com objetos físicos, altíssima heterogeneidade dos dispositivos, grande dinamismo, necessidade de ciência de contexto e restrição de recursos.

Sistemas de DTs apresentam requisitos que extrapolam os tradicionalmente encontrados em aplicações de IoT. Enquanto um *middleware* para IoT, em geral, foca na conectividade de dispositivos e no transporte de dados, um *middleware* para DT precisa tratar aspectos como semântica, modelagem de estado e sincronização contínua entre os mundos físico e o virtual, a chamada conexão causal. Além da simples coleta, transporte e processamento de dados de sensores, esses sistemas devem manter uma representação digital (instância de DT) continuamente sincronizada com o mundo físico, permitindo a observação, a análise e, em muitos casos, a atuação sobre as entidades representadas. Em outras palavras, enquanto um *middleware* convencional concentra-se primordialmente

na troca de mensagens e na abstração de heterogeneidade de hardware e de protocolos, um *middleware* para DT necessita dar suporte ao ciclo de vida integral da representação virtual. Isso implica garantir a sincronização bidirecional persistente e a consistência de estado entre a entidade do mundo real e sua contraparte digital, além de lidar com os demais requisitos já apresentados. Portanto, para tratar toda essa complexidade de forma sistemática, a literatura e as iniciativas industriais têm convergido para o uso de arquiteturas de *middleware* especializadas em sistemas de DTs.

Esta seção discute como soluções de *middleware* podem apoiar a construção de sistemas de DTs, descrevendo brevemente alguns exemplos da literatura e da indústria, tanto comerciais quanto de código aberto. O intuito é ilustrar que tais soluções permitem separar responsabilidades, promover interoperabilidade e facilitar a construção e a evolução de sistemas de DTs, ao atender parte ou todos os requisitos de tais sistemas ao longo de seu ciclo de vida.

5.3.1. Requisitos Atendidos por Middleware para Gêmeos Digitais

Integração. Como visto na Seção 5.2, um requisito essencial para sistemas de DTs refere-se à integração em diferentes níveis (entre entidades de DT, instâncias de DT e com sistemas externos), ao qual qualquer solução de *middleware* para DT deve atender. Diversas plataformas de *middleware* existentes utilizam a estratégia de desenvolver *drivers* específicos para permitir a comunicação tanto com ativos físicos quanto entre DTs [Redelinghuys et al. 2020, Barbie et al. 2024]. Nesse contexto, a interoperabilidade entre diferentes protocolos exige a implementação de manipuladores dedicados para cada protocolo, o que aumenta a complexidade e dificulta a escalabilidade das soluções. Outras propostas utilizam o protocolo MQTT e argumentam que o próprio *broker* de mensagens, componente central em arquiteturas baseadas nesse protocolo, permite a intercomunicação de DTs.

Há ainda abordagens que incorporam *frameworks* ou *middleware* de comunicação existentes [Lin and Lu 2024], como o serviço de distribuição de dados (DDS), a uma arquitetura mais completa de *middleware* para DT. Grande parte das soluções adota arquiteturas baseadas no modelo publicação-subscrição, devido à sua capacidade de desacoplar produtores e consumidores de dados, permitindo comunicação assíncrona e escalável entre componentes. Esse modelo facilita a interoperabilidade em ambientes heterogêneos, reduz a complexidade de integração e possibilita a evolução dinâmica do sistema, características essenciais para cenários com múltiplos DTs e as entidades do mundo real distribuídas.

Serviços de Dados. Serviços de dados também são centrais para sistemas de DTs e providos pela maior parte das soluções de *middleware*. Eles abrangem requisitos de representação e modelagem de DTs, armazenamento, aquisição e ingestão de dados, e processamento em tempo real. Soluções existentes adotam diferentes estratégias para representar DTs, relacionadas a metadados, identidade, atributos (estáticos ou dinâmicos), informações derivadas e relações entre DTs. Quanto à aquisição e ingestão de dados, em um ambiente onde o número de dispositivos e dados gerados pode aumentar rapidamente, recursos de coleta e ingestão devem estar presentes juntamente com o gerenciamento de fluxos de dados e processamento em lote. Algumas propostas utilizam soluções baseadas

em *brokers* de mensagens para lidar com a ingestão de dados, enquanto outras utilizam *brokers* para o armazenamento de médio prazo dos dados coletados. Para a persistência de dados a longo prazo, diferentes estilos de bancos de dados, incluindo abordagens na nuvem, têm sido empregados.

Inteligência. Com relação aos serviços de inteligência, um dos principais usos de DTs é imitar processos do mundo real por meio de simulações, e técnicas de inteligência computacional e análise de dados podem enriquecer a contraparte virtual desses processos. Entre outros objetivos, a incorporação de técnicas de aprendizado de máquina a sistemas de DTs permite prever falhas em ativos físicos, enquanto a análise de dados pode apoiar a tomada de decisões em tempo real, incluindo cenários de simulação com dados ingeridos em tempo real. Todavia, apesar de sua relevância e da forte integração entre aprendizado de máquina e DT, poucas propostas existentes tratam dessa integração em nível de *middleware*.

Gerenciamento. Requisitos de gerenciamento relacionam-se a funcionalidades de monitoramento do sistema, registro de *logs*, governança de dados e ao gerenciamento e à orquestração de recursos. A capacidade de monitorar continuamente o ambiente e seus elementos é crucial para garantir o funcionamento adequado de um DT. Por meio do monitoramento e de alertas do sistema, é possível observar DTs, aplicações e serviços, coletando, analisando e agindo sobre os dados para maximizar sua disponibilidade e desempenho. Além disso, a implementação de estratégias de registro de *logs* pode fornecer dados essenciais para análise e geração de melhorias adicionais para o sistema.

Governança. Outra capacidade de gerenciamento está relacionada à governança de dados, que envolve o gerenciamento da disponibilidade, usabilidade, integridade e segurança dos dados em DTs, com base em padrões e políticas que controlam seu uso. Poucas propostas de *middleware* na literatura incluem soluções completas de governança de dados. Na literatura, observa-se que a governança de dados pode ser facilitada isolando bancos de dados em microsserviços e utilizando *frameworks* e ferramentas de observabilidade para ajudar a garantir a disponibilidade, usabilidade, integridade e segurança dos dados.

Outras propostas adotam abordagens de sistemas de gerenciamento de banco de dados (SGBD) para armazenar os dados coletados de ativos físicos e aplicações externas e prover funcionalidades centralizadas de governança, como a rastreabilidade de dados. A orquestração é outra funcionalidade relacionada ao gerenciamento que o *middleware* pode fornecer para coordenar interações e fluxos de dados entre os diversos elementos de um sistema de DT, os quais podem variar de ativos físicos a aplicações corporativas. Mecanismos de registro e descoberta são igualmente essenciais para viabilizar estratégias de orquestração adequadas em DTs. Nesse contexto, muitos *middleware* adotam processos manuais para registrar um DT, fornecendo *endpoints* de API RESTful específicos.

Confiabilidade. Considerando a confiabilidade, requisitos de segurança, privacidade e criptografia de dados são essenciais para sistemas de DTs, principalmente em domínios de aplicação críticos em que têm sido utilizados. No entanto, ainda se observa uma carência de soluções voltadas para esse domínio e da incorporação de recursos de segurança como componentes essenciais de *middleware* para DT.

A seguir, descrevemos exemplos de *middleware* para DT que atendem a um ou mais desses requisitos. Observa-se que nenhuma plataforma existente atende a todos os requisitos simultaneamente. Além disso, há muitas soluções de *middleware* voltadas para domínios específicos de aplicação de DT. Na Seção 5.3.2 ilustramos exemplos de plataformas da indústria e na Seção 5.3.3 exemplos de soluções reportadas na literatura.

5.3.2. Iniciativas Proprietárias e de Código Aberto

***Azure Digital Twins* (Microsoft).** A *Azure Digital Twins* (ADT)[2] é uma plataforma como serviço (do inglês, *platform-as-a-service* – PaaS) que permite a criação de grafos de DTs baseados em modelos digitais de ambientes inteiros, tais como edifícios, fábricas, fazendas, redes de energia, ferrovias, estádios ou mesmo cidades. Esses modelos digitais podem ser usados para obter *insights* úteis para a melhoria de produtos, otimizações de operações, redução de custos e para proporcionar experiências inovadoras para as pessoas usuárias. Com foco em integração, a ADT pode ser usada para projetar uma arquitetura de sistemas de DTs na qual dispositivos físicos de IoT são representados no contexto de uma solução mais ampla, baseada na nuvem. Um importante diferencial da ADT é sua capacidade de integração nativa com o *Azure IoT Hub*[3], um serviço que funciona como um *hub* central de mensagens para soluções de IoT baseadas em nuvem, permitindo comunicação confiável e segura em larga escala entre aplicações de IoT e seus dispositivos conectados.

Na ADT, é possível definir entidades de DT que representam pessoas, lugares e objetos em um ambiente físico desejado usando tipos customizados de DTs, denominados modelos. Tais definições de modelos atuam como um vocabulário especializado para descrever um dado domínio de negócio. Modelos são definidos em uma linguagem similar a JSON, a *Digital Twins Definition Language* (DTDL)[4]. Modelos DTDL descrevem tipos de entidades de acordo com suas propriedades de estado, componentes e relacionamentos. Uma vez definidos os modelos, eles podem ser usados para criar gêmeos digitais (instâncias) que representem cada entidade específica no ambiente. Por exemplo, pode-se usar a definição de um modelo "Edifício" para criar vários DTs do tipo "Edifício" (por exemplo, "Edifício 1", "Edifício 2" e assim por diante). Podem-se também usar relacionamentos nas definições de modelos para conectar DTs entre si, formando um grafo conceitual. É importante destacar que tais modelos digitais, uma vez instanciados, são representações dinâmicas continuamente sincronizadas com o mundo real. A Figura 5.1 ilustra um modelo DTDL e uma instância correspondente, bem como o grafo de representação dos elementos do DT.

Para manter as propriedades da instância de DT atualizadas em relação ao ambiente físico, pode-se usar o *Azure IoT Hub*. Os dispositivos gerenciados pelo *hub* são representados como parte do grafo de DTs e fornecem os dados que mantêm o modelo atualizado. Pode-se criar um novo *hub* IoT para usar com a ADT ou conectar um *hub* existente juntamente com os dispositivos que ele já gerencia. Pode-se ainda alimentar a ADT a partir de outras fontes de dados, usando APIs RESTful ou conectores para outros serviços da plataforma Azure. A ADT fornece também um sistema de eventos robusto

[2]https://azure.microsoft.com/en-us/products/digital-twins

[3]https://azure.microsoft.com/en-us/products/iot-hub

[4]https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDL/v3/DTDL.v3.md

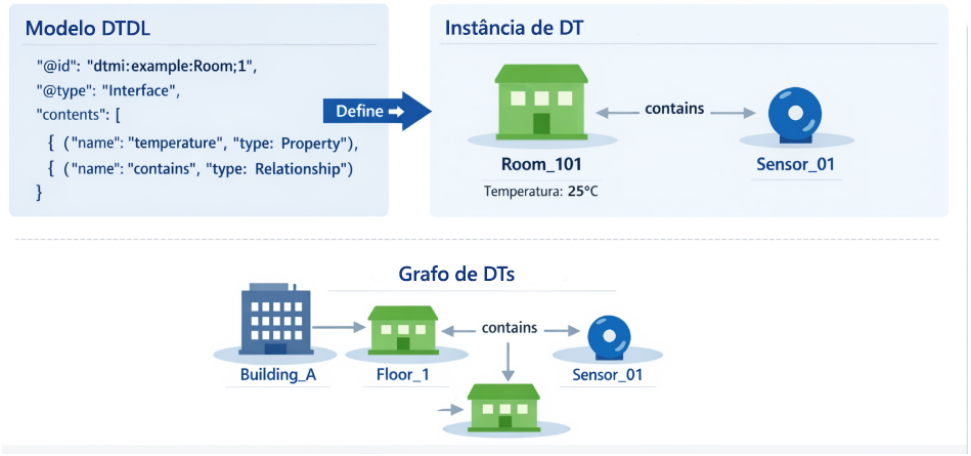


Figura 5.1. Exemplo de um modelo DTDL, de uma instância de DT e de um grafo de DTs

para manter o grafo atualizado, incluindo processamento de dados que pode ser personalizado para corresponder a cada lógica de negócios específica.

A ADT também oferece uma API de consulta que possibilita extrair *insights* do ambiente de execução em tempo real. Essa API permite realizar consultas com critérios de pesquisa abrangentes, incluindo valores de propriedades, relacionamentos, propriedades de relacionamento, informações do modelo, entre outras. Outra funcionalidade disponível é a visualização de grafos de DTs usando o *Azure Digital Twins Explorer*[5], que fornece uma interface para a construção e interação com o grafo conceitual. Por fim, o *Azure Digital Twins 3D Scenes Studio*[6] é um ambiente visual 3D imersivo em que pessoas usuárias podem monitorar, diagnosticar e investigar dados operacionais de DTs com o contexto visual de ativos 3D. Com um grafo de DTs e um modelo 3D, especialistas no domínio podem usar o componente construtor do estúdio para mapear os elementos 3D para DTs no grafo da ADT e definir a interatividade da interface do usuário e a lógica de negócios para uma visualização 3D de um ambiente de negócios. As cenas 3D podem então ser consumidas no *Azure Digital Twins 3D Scenes Studio* hospedado ou em uma aplicação personalizada que utiliza o componente de visualização 3D embarcado.

Eclipse BaSyx. A concretização da Indústria 4.0 demanda uma interoperabilidade em nível semântico que permita aos sistemas industriais interpretarem, de forma autônoma, as capacidades e o estado de cada elemento da cadeia produtiva. Nesse contexto, o *Asset Administration Shell* (AAS)[7] tem se estabelecido como a especificação técnica padronizada para a representação digital de um ativo. Concebido para solucionar o problema da heterogeneidade de protocolos no contexto da manufatura, o AAS atua como uma interface digital universal que abstrai a complexidade técnica dos equipamentos físicos e permite sua integração direta em ecossistemas conectados [Bader et al. 2022]. Para operacionalizar essa integração, um I4.0 *Component* é definido como a agregação lógica de dois elementos distintos e complementares: (i) o *Asset*, o qual se refere a qualquer entidade física

[5]https://learn.microsoft.com/en-us/azure/digital-twins/concepts-azure-digital-twins-explorer
[6]https://learn.microsoft.com/en-us/azure/digital-twins/quickstart-3d-scenes-studio
[7]https://reference.opcfoundation.org/I4AAS/v100/docs/4.1

ou lógica que possua valor para a organização (por exemplo, um motor, um sensor, um plano de produção ou um software), e; (ii) o AAS propriamente dito, que constitui a representação virtual que encapsula o *asset* e provê uma interface padronizada para o gerenciamento de suas informações e funcionalidades. Dessa forma, o AAS atua como a camada digital que permite ao *asset* físico ser reconhecido, conectado e gerenciado dentro do ecossistema da Indústria 4.0.

O Eclipse BaSyx[8] fornece um conjunto de componentes de software projetados para materializar o conceito de AAS em ambientes de produção, podendo, portanto, ser considerado um *middleware* para sistemas de DTs. Essa plataforma foi originalmente concebido pelo Fraunhofer IESE, da Alemanha, e atualmente é mantido pela Eclipse Foundation. O Eclipse BaSyx adota uma arquitetura orientada a serviços e microsserviços, composta por módulos independentes que interagem por meio de APIs padronizadas para viabilizar a persistência, a localização e a integração dos gêmeos digitais. A Figura 5.2 ilustra uma visão geral do Eclipse BaSyx. A seguir, descrevemos brevemente os componentes arquiteturais fundamentais que compõem o ecossistema do Eclipse BaSyx e suas respectivas responsabilidades.

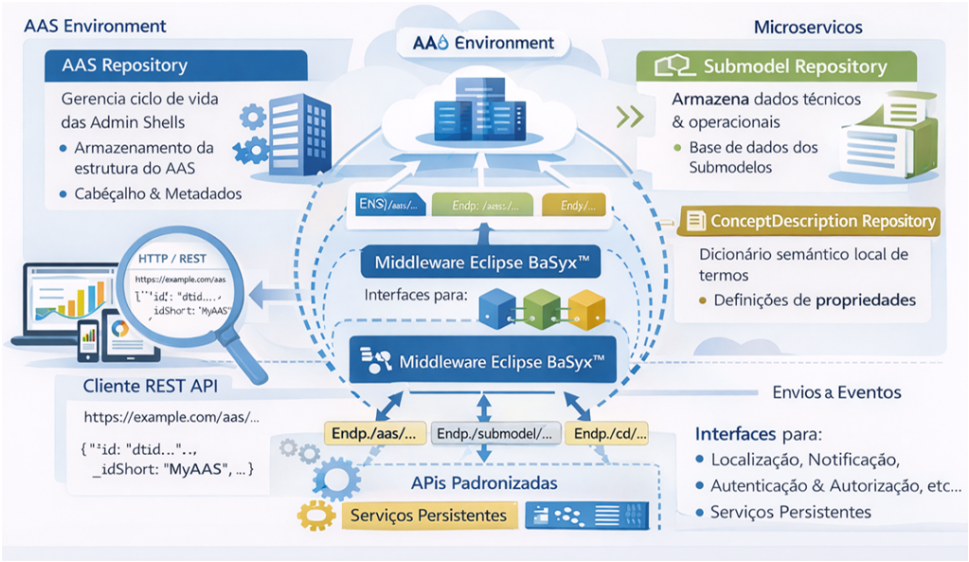


Figura 5.2. Visão geral do Eclipse BaSyx

AAS Environment: Atua como o servidor de aplicação principal da arquitetura, sendo responsável pela hospedagem e persistência de DTs. Ele consiste na agregação lógica de três interfaces de repositório distintas, acessíveis via protocolo HTTP/REST: *AAS Repository*, *Submodel Repository* e *ConceptDescription Repository*. O *AAS Repository* gerencia o ciclo de vida das entidades, armazenando a estrutura do AAS e permitindo operações de criação, leitura, atualização e remoção sobre o cabeçalho e os metadados do DT. O *Submodel Repository* é responsável pelo armazenamento efetivo dos dados técnicos e operacionais, gerenciando os submodelos e seus elementos internos e atuando como o banco de dados onde as propriedades estáticas e dinâmicas do ativo residem. O *ConceptDescription Repository* gerencia os dicionários semânticos locais, armazenando

[8]https://basyx.org

as definições que atribuem significado às propriedades (como unidades de medida e referências normativas) e permitindo que sistemas clientes consultem o significado exato de cada dado armazenado.

AAS Registry: Atua como um diretório de endereços do sistema, mantendo um mapeamento entre o identificador único da entidade de DT e o seu endereço de rede (*endpoint*). Dessa forma, aplicações clientes consultam o *AAS Registry* para resolver a localização de um AAS específico antes de estabelecer conexão direta com o repositório que o hospeda.

AAS Discovery Service: Complementar ao *AAS Registry*, oferece mecanismos de busca baseados em atributos. Enquanto o *AAS Registry* exige o conhecimento prévio do identificador da entidade para localizá-la, o *AAS Discovery Service* permite que aplicações pesquisem por DTs que possuam características específicas, por exemplo, listar todas as entidades de DT que contenham um Submodelo de "Consumo Energético". Esta funcionalidade viabiliza a descoberta dinâmica de recursos na rede industrial sem a necessidade de um inventário estático de identificadores.

DataBridge: Responsável pela conexão entre o ambiente virtual e os ativos físicos, ou seja, entre entidades e instâncias de DT. Ele atua como um *gateway* de tradução de protocolos, projetado para integrar equipamentos que não possuem suporte nativo ao padrão AAS. Sua operação baseia-se na configuração de rotas de sincronização que mapeiam sinais de protocolos industriais (como MQTT, OPC UA, HTTP ou Modbus) para propriedades específicas dentro dos submodelos. Esta operação ocorre em dois fluxos distintos. No sentido do físico para o digital, o componente monitora uma fonte de dados externa, como um tópico MQTT, e atualiza automaticamente a propriedade correspondente no servidor AAS sempre que um novo valor é recebido. No sentido do digital para o físico, ele observa alterações em uma propriedade do submodelo e propaga o novo valor para o dispositivo físico, permitindo o envio de comandos para atuadores.

BaSyxWeb User Interface: Atua como um painel de visualização genérico, permitindo que operadores e gestores inspecionem o estado dos ativos e naveguem pela estrutura hierárquica dos modelos AAS sem a necessidade de manipular diretamente chamadas de API ou *scripts* de código. O funcionamento da interface baseia-se na integração dinâmica com os componentes de registro e persistência. Ao ser inicializada, a aplicação conecta-se ao *AAS Registry* para recuperar a lista de DTs disponíveis na rede industrial. Quando um ativo é selecionado, a interface consulta automaticamente o *AAS Environment* correspondente para carregar e renderizar os submodelos e suas propriedades. Além da visualização passiva de telemetria, a ferramenta oferece suporte a operações de interação ativa, permitindo que pessoas usuárias autorizadas modifiquem valores de propriedades configuráveis ou acionem operações técnicas diretamente pelo navegador.

Eclipse Ditto. O Eclipse Ditto[9] é uma tecnologia de código aberto que tem se destacado como uma das principais referências para a implementação de DTs no contexto da IoT. Ele fornece representações digitais, denominadas *things*, de dispositivos físicos conectados, com suporte à sua modelagem, controle e monitoramento. A plataforma expõe APIs e protocolos Web e possui integração com módulos de fila e mensageria compatíveis com

[9]https://github.com/eclipse-ditto/ditto

diversos modelos amplamente difundidos, permitindo que os DTs sejam manipulados como serviços comuns e abstraindo complexidades de conectividade e infraestrutura.

A arquitetura do Eclipse Ditto é concebida segundo princípios modernos de engenharia de software, adotando o modelo hexagonal, no qual a lógica de domínio é isolada por meio de portas e adaptadores que intermediam as interações com o ambiente externo, promovendo modularidade e desacoplamento entre os componentes do sistema. A plataforma segue ainda uma abordagem baseada em microsserviços, distribuindo responsabilidades entre unidades especializadas que operam de forma autônoma e assíncrona. Além disso, o Eclipse Ditto emprega JSON como linguagem de interface entre artefatos e adota padrões como *Command Query Responsibility Segregation* (CQRS) e *event sourcing*, garantindo interoperabilidade, escalabilidade e desacoplamento para soluções distribuídas e heterogêneas baseadas em DTs.

O Eclipse Ditto implementa os seguintes serviços, conforme ilustra a Figura 5.3:

- *Gateway:* Atua como ponto de entrada principal, expondo interfaces REST e WebSocket para consumidores externos, além de realizar a autenticação inicial e o roteamento interno das mensagens em formato JSON.
- *Things:* É o núcleo funcional responsável pelo ciclo de vida dos DTs, mantendo estados, aplicando comandos e emitindo eventos persistidos e distribuídos aos serviços interessados.
- *Policies:* Gerencia regras de autorização, associando permissões específicas a identidades (pessoas usuárias, sistemas ou dispositivos) sobre os recursos.
- *Things Search:* Indexa e projeta eventos do serviço de *Things*, otimizando consultas complexas sobre atributos e metadados das entidades.
- *Connectivity:* Permite a integração com sistemas e dispositivos externos, oferecendo suporte nativo a diversos protocolos, como MQTT, AMQP e HTTP.

Figura 5.3. Serviços do Eclipse Ditto

No Eclipse Ditto, a comunicação consiste no mecanismo que assegura a coerência entre o mundo físico e sua representação digital. É por meio da troca de mensagens que telemetrias são incorporadas ao DT, comandos são propagados aos dispositivos físicos e o sistema mantém tanto reatividade em tempo real quanto consistência histórica. Para viabilizar essa comunicação de forma unificada e independente do protocolo de transporte, o Eclipse Ditto adota uma padronização semântica baseada em um modelo de mensagens estruturadas, garantindo interoperabilidade entre componentes heterogêneos. Essa abordagem permite que diferentes meios de comunicação coexistam sob um mesmo modelo

lógico, além de prover mecanismos para assegurar a integridade dos dados, a robustez diante de desconexões e o atendimento a requisitos de baixa latência quando necessários. Todos esses objetivos são atingidos por meio do *Ditto Protocol*.[10], que é uma especificação semântica baseada em JSON para estruturar e interpretar comandos, eventos e interações sobre *things*, de forma independente do protocolo de transporte adotado. Sua adoção viabiliza uma representação padronizada para operações como leitura, modificação e exclusão de propriedades, bem como para o envio de comandos a dispositivos físicos, garantindo uniformidade semântica em toda a arquitetura.

Um aspecto central do *Ditto Protocol* é a distinção entre dois canais de comunicação complementares. O canal denominado *Twin* é responsável por acessar e manipular o estado persistente da *thing*, isto é, o modelo digital armazenado na plataforma. As informações lidas ou alteradas por meio desse canal permanecem dissociadas da conectividade imediata com o dispositivo físico, permitindo a manutenção de um espelho confiável e histórico do estado do artefato. Já o canal denominado *Live* estabelece a comunicação em tempo real com o dispositivo físico, possibilitando o envio direto de comandos e a recepção instantânea de eventos. Por operar de forma síncrona com o mundo físico quando há conectividade ativa, este canal é indispensável em cenários que exigem baixa latência e controle imediato. Ao integrar esses dois canais em uma única especificação semântica, o *Ditto Protocol* garante que o modelo digital possa simultaneamente preservar estados históricos e responder de forma imediata a interações, assegurando consistência, rastreabilidade e flexibilidade operacional. A Figura 5.4 ilustra esses conceitos.

Figura 5.4. Canais *Twin* e *Live* relacionados a *Thing* no contexto do Eclipse Ditto

Comparando ADT, Eclipse Basyx e Eclipse Ditto. Os três exemplos de *middleware* discutidos anteriormente destacam-se como plataformas consolidadas voltadas para sistemas de DTs, sendo o Eclipse BaSyx e o Eclipse Ditto soluções de código aberto e a ADT uma solução proprietária baseada em nuvem. A seguir, traçamos um breve comparativo entre as três plataformas em termos dos seguintes aspectos:

- **Estratégia de sincronização:** O Eclipse Ditto destaca-se pela separação clara entre o canal *Twin* (acesso ao estado histórico e persistente) e o canal *Live* (interação direta e síncrona com o dispositivo), o que garante reatividade e consistência

[10]https://eclipse.dev/ditto/protocol-overview.html

histórica simultaneamente. Já o Eclipse BaSyx foca na tradução de protocolos de equipamentos para atualizar as propriedades técnicas dos submodelos do AAS.

- **Modelo de mensageria:** O Eclipse Ditto adota um protocolo semântico próprio, o *Ditto Protocol*, operando sobre uma arquitetura de microsserviços e mensageria orientada a eventos. A ADT utiliza um sistema de eventos robusto para manter seu grafo de representação atualizado conforme a lógica de negócio.
- **Segurança e acesso:** O Eclipse Ditto oferece controle de autorização granular por meio do serviço *Policies*, permitindo configurar permissões detalhadas de leitura e escrita para pessoas usuárias ou sistemas sobre recursos específicos. O Eclipse BaSyx, por sua vez, delega essas funções para provedores de identidade externos, como OAuth2.
- **Escopo e modelagem de dados:** A ADT adota a linguagem DTDL, baseada em JSON, enquanto o Eclipse Ditto define modelos de *things* também via JSON. Já o Eclipse BaSyx adota o AAS como padrão de modelagem. Enquanto a ADT é mais adequada para visualizar e consultar grandes grafos de relacionamentos, o Eclipse BaSyx é o mais rigoroso na definição semântica industrial em termos de dicionários semânticos e unidades de medida normatizadas. O Eclipse Ditto, por sua vez, foca na abstração da complexidade do hardware, permitindo que as instâncias de DTs sejam manipulados como serviços Web comuns.
- **Foco de aplicação:** A ADT é adequada para representar, por meio de grafos, ambientes complexos, como cidades e redes de energia. O Eclipse Ditto foi concebido com foco em ambientes de IoT, enquanto o Eclipse BaSyx é voltado para ambientes fabris e para lidar com o desafio de interoperabilidade na Indústria 4.0.

5.3.3. Propostas da Literatura

Considerando o estado da arte atual, a pesquisa sobre *middleware* para DTs ainda é relativamente recente, considerando que a necessidade de soluções especificamente projetadas para esses ambientes passou a receber maior atenção apenas nos últimos anos. Em geral, as abordagens existentes concentram-se em aspectos pontuais, como protocolos de comunicação, padrões de integração industrial ou mecanismos de sincronização, mas frequentemente carecem de uma plataforma unificada capaz de abordar, de forma integrada, questões de modelagem, orquestração, integração e execução. Além disso, muitas soluções são direcionadas a domínios específicos de aplicação, o que limita sua generalização para diferentes cenários de sistemas de DTs. A seguir, apresentamos algumas propostas representativas da literatura, destacando como elas atendem aos requisitos previamente analisados.

Tao e Zhang [Tao and Zhang 2017] propõem um *middleware* para DTs voltado a ambientes fabris no contexto da Indústria 4.0. O objetivo principal do *middleware* é utilizar dados provenientes de sensores para realizar simulações em tempo real da linha de montagem, permitindo o ajuste de parâmetros de produção sem a necessidade de interromper as máquinas. O *middleware* é responsável pela chamada Unidade de Processamento de Dados do DT, que executa funções como limpeza, compressão e fusão de dados oriundos de sensores industriais, além de atualizar, quase instantaneamente, modelos 3D e de simulação com base nos dados físicos.

Mahdi et al. [Mahdi et al. 2025] apresentam uma arquitetura baseada em DTs específica para aplicações do tipo *Wire Arc Additive Manufacturing* (WAAM). Nesse contexto, o DT consiste em um conjunto de ferramentas voltadas ao aumento da precisão, eficiência e garantia de qualidade do processo produtivo. A arquitetura incorpora redes neurais convolucionais (CNNs) para detecção de anomalias, recursos de visualização 3D para representação detalhada do processo e técnicas de visão computacional para identificação precisa de parâmetros das peças. Os autores destacam que a implementação de DTs nesse cenário exige o tratamento de grandes volumes de dados, provenientes de múltiplas fontes, como sensores inteligentes, imagens, sinais analógicos e digitais e registros de inventário. Para viabilizar a comunicação entre entidades físicas e seus respectivos DTs, é utilizado o padrão OPC UA [Zhang et al. 2023], que oferece interoperabilidade semântica, independência de plataforma e comunicação segura em tempo real. A arquitetura proposta combina OPC UA com técnicas de análise de dados e armazenamento em banco de dados não relacional em nuvem, oferecendo suporte a comunicação, gerenciamento de dados estruturados, segurança e controle em tempo real. Além disso, integra mecanismos avançados de visualização e predição de defeitos.

Kutz et al. [Kuts et al. 2019] exploram o uso de DTs no gerenciamento e controle de sistemas de produção, com ênfase em robótica industrial, sendo derivado da adaptação de uma solução previamente desenvolvida para a Internet Industrial das Coisas (IIoT) [Modoni et al. 2018]. A proposta incorpora o uso de realidade virtual para simular a presença humana, permitindo interação remota e colaborativa. Os autores identificam requisitos fundamentais para sistemas de DTs, incluindo aquisição de dados via sensores, configuração precisa do modelo digital, uso de modelos de dados representativos e sincronização entre os sistemas físico e digital. O foco principal do trabalho está na sincronização, sendo proposto um *middleware* orientado a mensagens baseado no modelo publicação-subscrição, responsável por propagar dados de telemetria entre os ativos físicos e suas representações digitais.

André et al. [André et al. 2020] investigam a separação de interesses relacionadas à comunicação em relação a outros aspectos da aplicação, com o objetivo de aumentar a flexibilidade, adaptabilidade e evolutividade dos sistemas de DT. O trabalho propõe um *framework* que atua como camada intermediária de comunicação entre sistemas de controle de manufatura e os componentes de fábrica, incluindo DTs e simuladores.

Yun et al. [Yun et al. 2017] apresentan uma *plataforma* de DT voltada a sistemas ciberfísicos críticos, como veículos autônomos com sistemas avançados de assistência à direção. A arquitetura proposta contempla um ambiente distribuído de cooperação entre DTs, um *middleware* de comunicação centrado em dados baseado no padrão DDS e suporte a co-simulação por meio do padrão *Functional Mock-Up Interface* (FMI)[11]. Os autores destacam a importância da interoperabilidade em tempo real e da distribuição eficiente de dados como requisitos centrais para sistemas de DTs em larga escala. Além disso, enfatizam a necessidade de suporte a requisitos de qualidade de serviço (QoS), sincronização temporal entre simuladores e gerenciamento de dados complexos e multimídia. Nesse contexto, o *middleware* desempenha papel fundamental não apenas na comunicação entre entidades físicas e digitais, mas também na integração entre diferentes

[11] https://opensimulationplatform.com/co-simulation/

sistemas e simuladores.

Por fim, Bellavista et al. [Bellavista et al. 2024] abordam os desafios relacionados à sincronização e à orquestração de DTs, introduzindo o conceito de *entanglement*, originalmente proposto no trabalho de Minerva et al. [Minerva et al. 2020]. Esse conceito representa o grau de alinhamento entre o DT e sua contraparte física, sendo considerado essencial para garantir a confiabilidade da representação digital. Com base nesse conceito, os autores propõem um *middleware entanglement-aware*, capaz de monitorar e manter o nível de sincronização entre os sistemas. A solução utiliza conteinerização e considera aspectos como proximidade física entre DT e ativo real, bem como a competição por recursos computacionais entre diferentes DTs. A arquitetura é composta por um nó central, responsável pelo controle global, e por nós trabalhadores, que executam os DTs em diferentes camadas, na borda ou na nuvem. O sistema inclui ainda interfaces para monitoramento, orquestração, gerenciamento e visualização. Ademais, a arquitetura dispõe de componentes para gerenciamento de dados, armazenando métricas, eventos, configurações e artefatos de DTs.

A Tabela 5.1 sintetiza a cobertura dos requisitos fundamentais para DTs pelas soluções analisadas com base nas categorias propostas pelo DTC. Como podemos observar, nenhuma das soluções analisadas atende completamente a todas as categorias. As categorias *Integração* e *Serviços de Dados* são cobertas de forma consistente, especialmente nas plataformas mais maduras. Por outro lado, as categorias *Inteligência* e *Confiabilidade* concentram as lacunas mais expressivas. Apesar da evolução, a integração de técnicas de aprendizado de máquina e análise avançada de dados ainda é tratada de forma periférica pela maioria das soluções, e os mecanismos de segurança, quando presentes, são geralmente provenientes de *frameworks* genéricos, sem especialização para o contexto de DTs. A categoria *Gerenciamento*, especialmente no tocante a orquestração e alocação de recursos, é atendida de forma satisfatória apenas por soluções que o elegem como foco central, como o caso do Eclipse BaSyx e da proposta de Bellavista et al.

Além da diferença de maturidade em relação a soluções proprietárias, é possível perceber que as atuais soluções de *middleware* para DTs abordam os requisitos de forma generalista, deixando lacunas em inteligência e segurança, ou são altamente especializadas para nichos específicos, atendendo bem a aspectos de um contexto industrial, por exemplo. Esses resultados reforçam que o desenvolvimento de *middleware* para DTs ainda é uma área em evolução, com espaço para contribuições que integrem as seis categorias, requisitos de alto nível e capacidades técnicas necessárias de forma mais abrangente.

5.4. Um Middleware para Sistemas de Gêmeos Digitais: Componentes e Implementação

A análise das soluções comerciais, de código aberto e das propostas da literatura, apresentada na Seção 5.3.3, evidencia que, apesar dos avanços significativos no desenvolvimento de *middleware* para sistemas de DTs, ainda persistem lacunas importantes, especialmente no que se refere à integração simultânea dos diferentes requisitos discutidos anteriormente neste capítulo. De modo geral, observa-se que essas soluções tendem a se concentrar em aspectos específicos, como integração de dispositivos, gerenciamento de dados ou mecanismos de comunicação, sem oferecer uma abordagem unificada que contemple, de forma

Tabela 5.1. Comparativo entre soluções de *middleware* para DTs

Requisitos / Soluções	Serviços de Dados	Integração	Inteligência
Microsoft ADT	✔	✔	✕
Eclipse BaSyx	✔	✔	✕
Eclipse Ditto	◯	✔	✕
[Yun et al. 2017]	◯	✔	✕
[Tao and Zhang 2017]	✔	✕	✕
[Kuts et al. 2019]	◯	✔	✕
[André et al. 2020]	✕	✔	✕
[Bellavista et al. 2024]	✔	✕	◯
[Mahdi et al. 2025]	✔	✔	✔

Requisitos / Soluções	Experiência de Usuário	Gerenciamento	Confiabilidade
Microsoft ADT	✔	◯	✕
Eclipse BaSyx	◯	✔	◯
Eclipse Ditto	✕	◯	◯
[Yun et al. 2017]	✕	✕	✕
[Tao and Zhang 2017]	✕	✕	✕
[Kuts et al. 2019]	✕	✕	✕
[André et al. 2020]	✕	✕	✕
[Bellavista et al. 2024]	◯	✔	✕
[Mahdi et al. 2025]	◯	✕	◯

✔ = requisito atendido; ◯ = requisito parcialmente atendido; ✕ = requisito não atendido

integrada, as dimensões de modelagem, orquestração, sincronização e operação dos DTs. Além disso, muitas propostas apresentam forte acoplamento a domínios específicos ou dependem de configurações complexas para garantir a interoperabilidade entre componentes heterogêneos.

Esta seção apresenta o MidDiTS [Pereira et al. 2024], um *middleware* concebido para apoiar, de forma integrada, o desenvolvimento e a operação de sistemas de DTs. A seguir, detalhamos sua arquitetura e seus principais componentes de implementação.

5.4.1. Arquitetura

A arquitetura do MidDiTS foi projetada para apoiar a modelagem, o desenvolvimento, a sincronização e a execução de sistemas de DTs. Sua concepção baseia-se na premissa de que o desenvolvimento de sistemas de DTs requer a abstração das complexidades de comunicação de baixo nível, o gerenciamento da heterogeneidade dos dispositivos e a garantia de consistência causal bidirecional entre os mundos físico e virtual.

Para fundamentar sua operação, o MidDiTS apoia-se em três conceitos principais: (i) a *entidade de DT*, que representa o ativo físico no mundo real; (ii) a *instância de DT*, que corresponde à contraparte digital em execução e continuamente sincronizada pelo *middleware*; e (iii) o *sistema de DT*, que denota a aplicação de mais alto nível responsável por agregar e orquestrar múltiplas instâncias interconectadas. Adicionalmente, o MidDiTS permite que múltiplas aplicações compartilhem a mesma infraestrutura subjacente, mantendo, contudo, o isolamento de seus modelos, regras de negócio e instâncias.

A Figura 5.5 apresenta a arquitetura do MidDiTS, cujas funcionalidades são organizadas em componentes lógicos distribuídos em quatro camadas principais: *Things Facade*, *Orchestration Layer*, *DT Gateway* e *Security Management*.

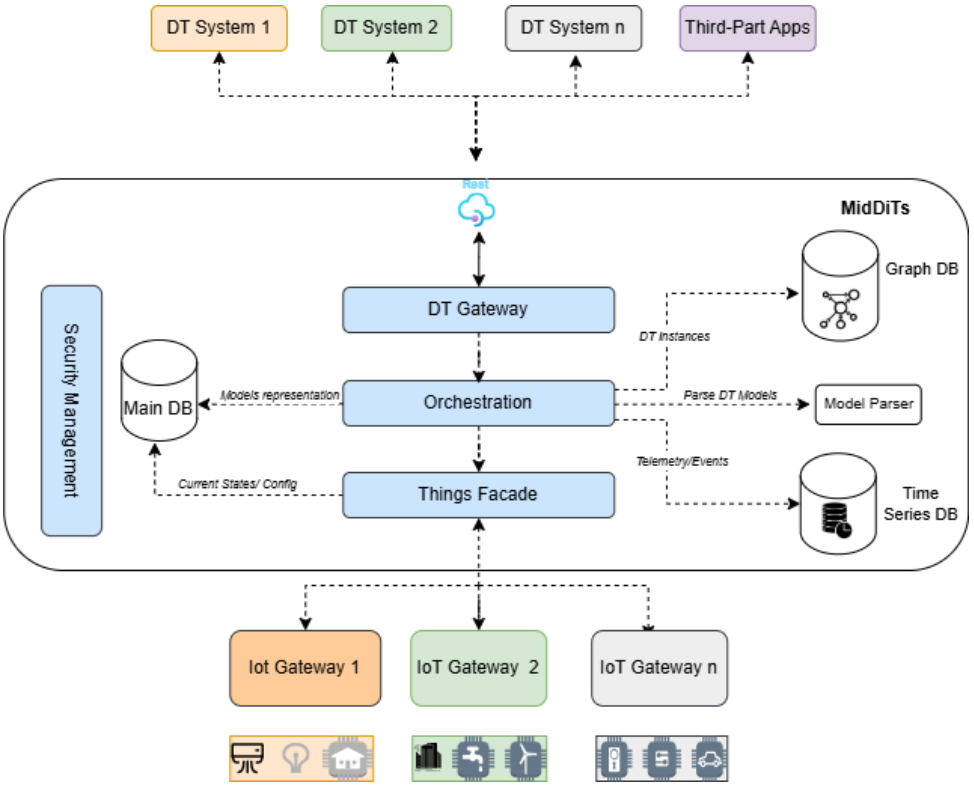


Figura 5.5. Visão geral da arquitetura do MidDiTS

A camada *Things Facade* atua como a interface de conexão entre o MidDiTS e as entidades físicas. Em vez de implementar diretamente protocolos de comunicação com dispositivos, essa camada integra-se a um *gateway* IoT externo. Sua principal função é mapear metadados de dispositivos (incluindo tipos e propriedades) para representações internas no *middleware*. Além disso, provê serviços de descoberta de dispositivos, gerencia o fluxo de telemetria e viabiliza o envio de comandos.

Devido ao fluxo contínuo e bidirecional de dados, a *Things Facade* requer suporte de sistemas de armazenamento otimizados para séries temporais, com o objetivo de registrar o histórico das propriedades e monitorar métricas como latência de comunicação. O MidDiTS incorpora ainda um gerenciador de eventos capaz de não apenas agendar requisições periódicas, mas também reagir dinamicamente a mudanças operacionais na infraestrutura.

O núcleo do MidDiTS é a *Orchestration Layer*. Sua principal responsabilidade é manter o estado e os modelos dos DTs continuamente sincronizados com suas contrapartes físicas. Essa camada armazena definições de modelos, gerencia a organização dos DTs por meio de relacionamentos dinâmicos entre instâncias e trata da conectividade por meio de mecanismos de *binding* explícitos e semânticos. Além disso, garante a propagação consistente de mudanças de estado para as aplicações consumidoras.

A interface de comunicação com aplicações externas é provida pelo *DT Gateway*. Baseado no padrão *API Gateway*, ele centraliza a exposição de serviços RESTful, gerenciando o roteamento e a composição de requisições, bem como a integração com aplicações de terceiros e sistemas de DTs independentes. O *gateway* oferece uma interface estável para acesso a dados agregados (por exemplo, retornando uma instância de DT juntamente com sua telemetria recente) e oferece suporte a interações síncronas e assíncronas, incluindo a emissão de identificadores para processamento em segundo plano.

De forma transversal, a camada *Security Management* garante o controle centralizado de autenticação e autorização. Considerando que sistemas de DTs frequentemente operam em ambientes críticos, essa camada gerencia políticas de acesso, emite credenciais e tokens de curta duração e assegura a comunicação segura entre o middleware, os *gateways* IoT e os clientes da API.

5.4.2. Implementação

No âmbito da implementação, todos os componentes do MidDiTS são empacotados como *containers*, o que facilita a implantação, a orquestração e o isolamento de dependências (ver Figura 5.6). A maior parte do *middleware* foi desenvolvida em Python, utilizando o *framework* Django. Para suportar as funcionalidades do *DT Gateway* e da camada de segurança, utiliza-se o nginx[12] como *proxy* reverso, balanceador de carga e mecanismo adicional de proteção, garantindo maior resiliência e escalabilidade.

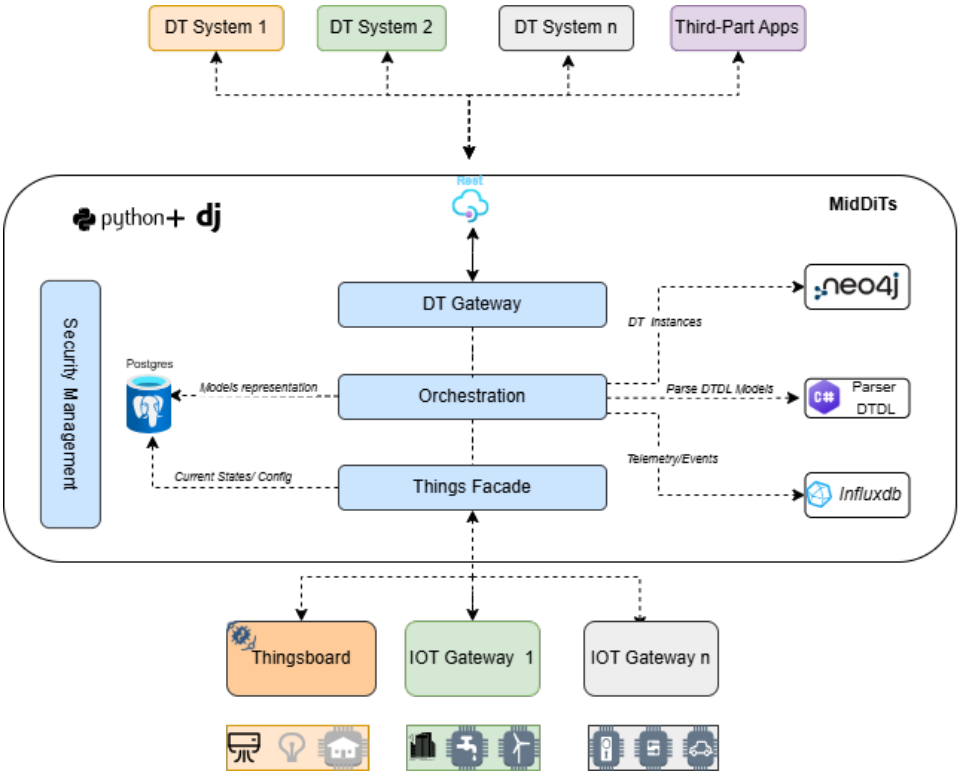


Figura 5.6. Visão geral da implementação atual da arquitetura do MidDiTS

[12]https://nginx.org

Para padronizar a definição das instâncias, o MidDiTS adota a DTDL, permitindo especificar interfaces, telemetrias, comandos e relacionamentos de ativos físicos. A conversão dessas definições para representações estruturadas utilizadas internamente é realizada por meio de um motor de *parsing* (*DTDL Parser*), desenvolvido pelo DTC[13], executado em *container* dedicado e acessado via REST.

A persistência de dados no MidDiTS reflete a heterogeneidade das informações manipuladas em um ecossistema de DTs, sendo implementada por meio da integração de três tecnologias distintas. O PostgreSQL é utilizado como repositório relacional para definições estruturais, modelos convertidos e dados administrativos. O Neo4j[14] é empregado para modelar a estrutura dinâmica do sistema como um grafo, permitindo consultas complexas e navegação eficiente entre instâncias e seus relacionamentos. Por sua vez, o InfluxDB[15] é utilizado para armazenamento de séries temporais, registrando dados de telemetria e permitindo análise de latência e sincronização.

A comunicação bidirecional com a contraparte física é delegada ao *ThingsBoard IoT Gateway*[16], que traduz comandos do *middleware* para protocolos nativos dos dispositivos e expõe o estado da infraestrutura física. Para evitar sobrecarga de rede associada a técnicas de *polling*, o MidDiTS adota um modelo orientado a eventos baseado em WebSockets, garantindo baixa latência na propagação de atualizações.

Para suprir limitações da modelagem com a DTDL no que diz respeito à sincronização bidirecional, o MidDiTS introduz a diretiva *Causal*. Quando aplicada a uma propriedade, essa diretiva ativa um mecanismo dedicado de sincronização, garantindo a propagação consistente de alterações entre os domínios físico e digital.

Por fim, o acoplamento entre propriedades do DT e sensores/atuadores do *gateway* IoT é realizado por meio do conceito de *binding*. Esse processo pode ser definido explicitamente pelo desenvolvedor ou automatizado por meio de um mecanismo de *binding* semântico, que utiliza técnicas de Inteligência Artificial para gerar representações vetoriais (*embeddings*) a partir de descrições textuais. A similaridade semântica é então calculada (por exemplo, via similaridade de cosseno), permitindo a inferência automática de correspondências entre elementos físicos e digitais. Essa abordagem reduz o esforço de configuração e melhora a integração em ambientes IoT heterogêneos e de larga escala.

5.5. Processo de Desenvolvimento de Sistemas de Gêmeos Digitais com o MidDiTS

Como detalhado na seção anterior, o MidDiTS atua como uma camada de abstração entre o mundo físico e o virtual. Sob a perspectiva de quem projeta sistemas de DTs, seu principal valor está em permitir que o desenvolvimento se concentre na lógica de negócio e nas aplicações finais, enquanto aspectos como comunicação de baixo nível, interoperabilidade e consistência de dados são gerenciados pelo *middleware*.

O processo de desenvolvimento de sistemas de DTs com o MidDiTS segue uma

[13]https://github.com/digitaltwinconsortium/DTDLParser
[14]https://neo4j.com/product/neo4j-graph-database/
[15]https://www.influxdata.com/products/influxdb/
[16]https://thingsboard.io/docs/iot-gateway/

abordagem estruturada em quatro fases principais (ver Figura 5.7): (i) *Projeto*, responsável pela definição dos modelos; (ii) *Configuração*, na qual o ambiente é preparado e integrado aos dispositivos físicos; (iii) *Desenvolvimento*, que materializa instâncias e relacionamentos; e (iv) *Operação*, que garante a execução contínua e sincronizada do sistema. A seguir, cada uma dessas fases é detalhada a partir do cenário de um condomínio inteligente, baseado na *Smart Cities Ontology*[17], composto por casas, cômodos, sensores e atuadores interconectados, conforme ilustrado na Figura 5.8.

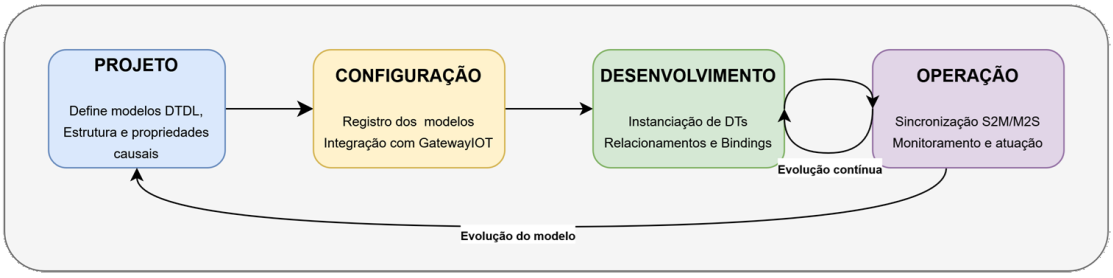


Figura 5.7. Processo de desenvolvimento de sistemas de DTs com o MidDiTS

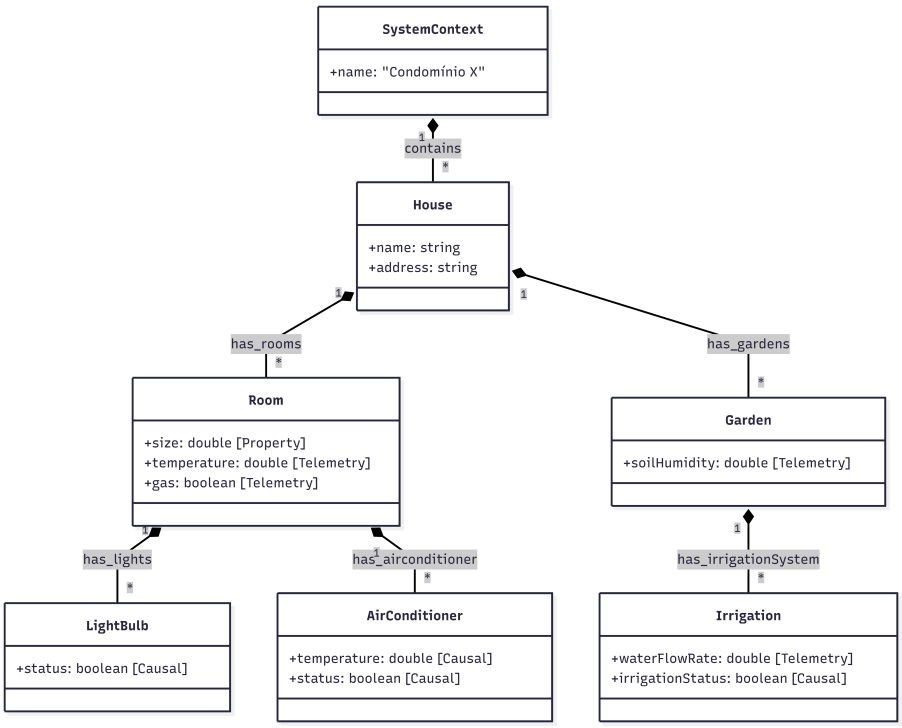


Figura 5.8. Condomínio inteligente baseado na *Smart Cities Ontology*

5.5.1. Projeto – Modelagem com DTDL

A fase de projeto estabelece a base conceitual de um ecossistema de DTs. Nela, o desenvolvedor define as representações digitais dos ativos físicos utilizando a linguagem

[17] https://github.com/Azure/opendigitaltwins-smartcities

DTDL, que permite descrever, de forma padronizada e processável por máquina, propriedades, telemetrias, comandos e relacionamentos. De forma geral, o desenvolvedor deve estruturar os modelos considerando quatro dimensões principais:

- **Modelos estruturais e espaciais**: representam a organização lógica e física do ambiente, como *House*, *Room* e *Garden*.
- **Modelos de dispositivos**: representam sensores e atuadores conectados, como *LightBulb* e *AirConditioner*.
- **Propriedades e fluxos de dados**: descrevem atributos monitorados por sensores ou controlados por atuadores.
- **Relacionamentos**: definem como as entidades se conectam, formando a estrutura do DT.

Essa organização permite compreender o modelo como uma estrutura conectada ao ambiente físíco e não apenas como um conjunto isolado de atributos, refletindo assim tanto sua organização quanto seu comportamento. Com base nessa modelagem, o MidDiTS introduz extensões e mecanismos que enriquecem a semântica desses modelos, permitindo uma integração mais direta e automatizada entre o mundo físico e o virtual.

Telemetria vs. propriedades causais. Modelos tradicionais em DTDL contemplam a definição de propriedades, comandos e telemetrias, porém não explicitam, de forma semântica, quais propriedades devem manter sincronização ativa com o ambiente físico. Na prática, essa relação costuma ser delegada à lógica da aplicação, o que dificulta a padronização e a compreensão do comportamento do DT. O MidDiTS trata essa limitação ainda na fase de projeto ao introduzir *propriedades causais* como extensão à DTDL, permitindo declarar explicitamente quais atributos possuem relação direta com dispositivos físicos. Assim, uma telemetria representaria dados observacionais, sem garantia de sincronização contínua, enquanto uma propriedade causal seria um atributo com sincronização bidirecional consistente entre os mundos físicos e virtual. Essa abordagem fortalece a coerência do modelo e viabiliza a automação da sincronização.

Para habilitar esse comportamento, o desenvolvedor inclui a extensão causal `dtmi:dtdl:extension:causal` no início do arquivo JSON correspondente ao modelo. O Código 5.1 mostra um modelo DTDL de um sistema de irrigação. Nesse trecho de código, a extensão causal é declarada no atributo `@context` (linha 2), permitindo que determinadas propriedades do modelo sejam tratadas como causais. A telemetria denominada `waterFlowRate` (linhas 8 a 12) representa um fluxo de dados proveniente do mundo físico, sendo utilizada para o monitoramento do fluxo de água. Em contraste, a propriedade denominada `irrigationStatus` (linhas 15 a 19) é definida como uma propriedade causal, o que indica que seu valor está diretamente associado ao estado de um dispositivo físico, por exemplo, uma válvula de irrigação, permitindo não apenas a leitura, mas também a atuação sobre o ambiente. Dessa forma, alterações realizadas no modelo digital podem ser automaticamente propagadas para o dispositivo físico, enquanto mudanças no ambiente físico atualizam o estado do DT, caracterizando uma relação causal contínua entre os dois domínios.

Ontologia e tipos quantitativos. No nosso cenário do condomínio inteligente, a modelagem é fundamentada na *Smart Cities Ontology*. A partir dessa ontologia, entidades

```
{
    "@context": ["dtmi:dtdl:context;3","dtmi:dtdl:extension:causal;1"
        ↪ ],
    "@id": "dtmi:housegen:Irrigation;1",
    "@type": "Interface",
    "displayName": "Irrigation",
    "contents": [
        {
            "@type": "Telemetry",
            "name": "waterFlowRate",
            "schema": "double",
            "displayName": "Water Flow Rate",
            "description": "The rate of water flow in liters per
                ↪ minute"
        },
        {
            "@type": ["Property","Causal"],
            "name": "irrigationStatus",
            "schema": "boolean",
            "displayName": "Set the Irrigation status",
            "description": "Defines if the irrigation system is ON or
                ↪  OFF"
        }
    ]
}
```

Código 5.1. Modelo DTDL com extensão causal para sistema de irrigação

como *House*, *Room* ou *Garden* passam a ser tratadas como objetos espaciais (*SpatialObject*), permitindo a incorporação de conceitos geoespaciais por meio de padrões como o GeoSPARQL[18]. Elementos fundamentais como *Geometry* e *Feature* possibilitam representar não apenas a estrutura lógica do ambiente, mas também sua organização espacial, o que é essencial em cenários urbanos e de cidades inteligentes.

Para representar grandezas físicas de forma precisa, o MidDiTS utiliza a extensão de tipos quantitativos (`dtmi:dtdl:extension:quantitativeTypes`). Essa extensão permite que propriedades sejam associadas a unidades e classificações físicas padronizadas. Por exemplo, ao modelar um cômodo (*Room*), a propriedade *size* pode ser definida como uma área, medida em metros quadrados, enquanto sensores de temperatura podem ser explicitamente definidos considerando temperatura em graus Celsius. Isso reduz ambiguidades e facilita a interoperabilidade entre sistemas.

Definição da estrutura do sistema de DTs. Outro aspecto fundamental da fase de projeto é a definição explícita da estrutura do sistema de DTs, ou seja, como os diferentes componentes relacionam-se entre si. Essa estrutura é descrita diretamente nos modelos DTDL por meio do atributo `Relationship`. Por exemplo, o modelo *House* pode definir um relacionamento *has_rooms*, apontando para instâncias do modelo *Room*. Da mesma forma, o modelo *Room* pode estabelecer o relacionamento *has_lights* com uma multiplicidade definida para indicar a quantidade máxima de dispositivos associados. Essa definição não é apenas estrutural, mas também operacional, pois estabelece a forma como as in-

[18]https://www.ogc.org/standards/geosparql/

stâncias de DTs se conectam e interagem dentro do sistema. No MidDiTS, essas relações são utilizadas pela *Orchestration Layer* para materializar relacionamentos no banco de dados orientado a grafos (Neo4j), evidenciando o sistema de DTs como uma rede de entidades interconectadas. Dessa forma, decisões tomadas ainda na fase de modelagem impactam diretamente a forma como consultas, navegação e análises serão realizadas posteriormente.

Ao final da fase de projeto, o desenvolvedor terá em mãos um conjunto de modelos DTDL semanticamente enriquecidos, capazes de representar não apenas as propriedades e comportamentos dos ativos físicos, mas também suas relações estruturais, espaciais e causais. Esses modelos constituem a base para as etapas seguintes do processo de desenvolvimento de um sistema de DTs com o MidDiTS, permitindo que o *middleware* automatize a criação, sincronização e operação dos DTs no ambiente de execução.

5.5.2. Configuração

A fase de configuração no MidDiTS pode ser compreendida como um *pipeline* de ativação do sistema de DTs, no qual os modelos definidos na fase de projeto são progressivamente conectados ao ambiente de execução e aos serviços do *middleware*. Diferentemente da fase de projeto, que se concentra na definição estrutural e semântica dos DTs, a configuração estabelece as condições necessárias para que essas definições sejam efetivamente executadas, integrando modelos digitais, dispositivos físicos (ou simulados) e os mecanismos internos do *middleware*. Esse processo organiza-se em três etapas principais: (i) definição do contexto de execução e controle de acesso; (ii) integração com a infraestrutura física por meio da camada *Things Facade*; e (iii) preparação do ambiente para execução dos DTs. Cada uma dessas etapas é detalhada a seguir.

Definição do contexto de execução e controle de acesso. O processo de configuração inicia-se com a autenticação do desenvolvedor e a definição do contexto organizacional no qual os DTs serão gerenciados. Esse contexto estabelece o escopo de isolamento das aplicações, permitindo que diferentes sistemas de DTs coexistam na mesma infraestrutura de forma independente.

Nesse momento, são definidos elementos como organização, pessoas usuárias e papéis, gerenciados pela camada *Security Management*, responsável por garantir autenticação, autorização e controle de acesso às operações sobre os DTs. A Figura 5.9 ilustra o cadastro de uma organização e a associação de seus membros. Esse mecanismo garante o isolamento de contexto entre diferentes aplicações no MidDiTS.

Integração com a infraestrutura física (*Things Facade*). Uma vez estabelecido o contexto de execução, o próximo passo consiste na integração com o ambiente físico por meio do registro de um *gateway* IoT externo, como a *ThingsBoard*. Essa etapa ativa a camada *Things Facade*, responsável por abstrair os detalhes de comunicação com dispositivos físicos e consolidar suas representações no *middleware*.

O registro do *gateway* inclui a configuração de credenciais de acesso e a definição da URL do serviço. A Figura 5.10 apresenta um exemplo do cadastro de um *gateway* IoT no MidDiTS.

Após o registro, o MidDiTS realiza a descoberta automática de dispositivos disponí-

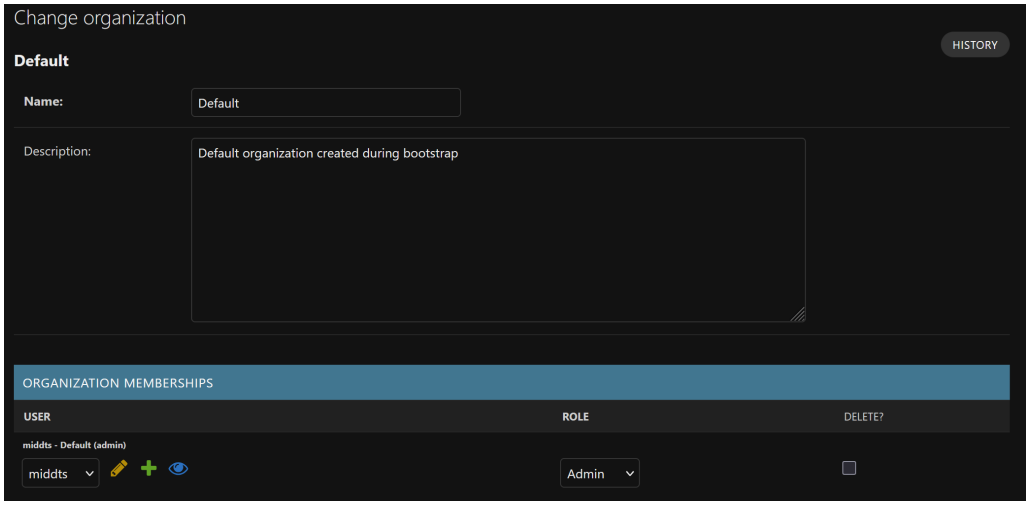


Figura 5.9. Cadastro de organização e gerenciamento de membros no MidDiTS

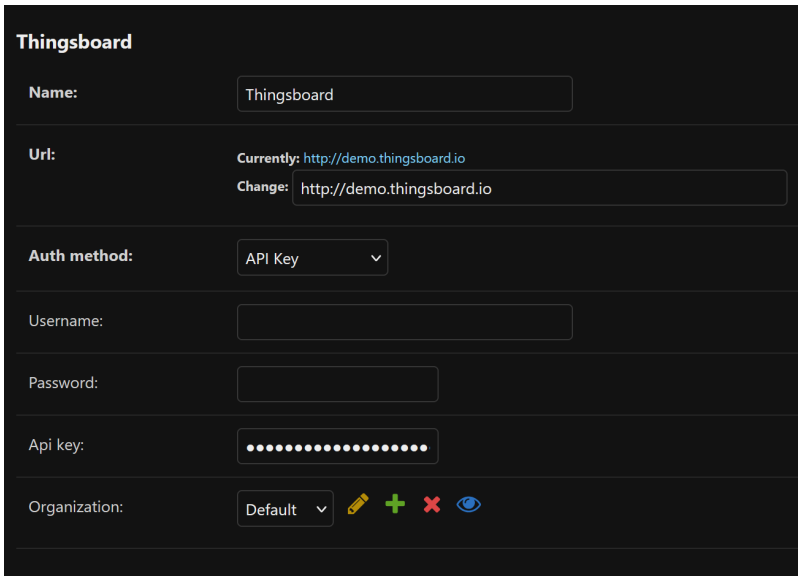


Figura 5.10. Cadastro de *gateway* IoT para integração com a *ThingsBoard*

veis no *gateway*. Esse processo consiste na consulta a APIs para obtenção de dispositivos, seus identificadores e metadados associados. Esses metadados incluem nomes, rótulos, atributos e chaves de telemetria, sendo fundamentais para a identificação e posterior associação com os elementos do modelo digital.

A Figura 5.11 ilustra o processo de descoberta de dispositivos. Nesse processo, o MidDiTS, por meio da camada *Things Facade*, realiza uma requisição ao *gateway* IoT para obtenção dos dispositivos cadastrados. Como resposta, o *gateway* retorna os metadados associados, que são então processados e armazenados pelo *middleware*, permitindo a construção de uma representação local dessas entidades.

Preparação do ambiente de execução dos DTs. Com a infraestrutura física integrada, a configuração avança para a preparação do ambiente de execução dos DTs. Nessa etapa,

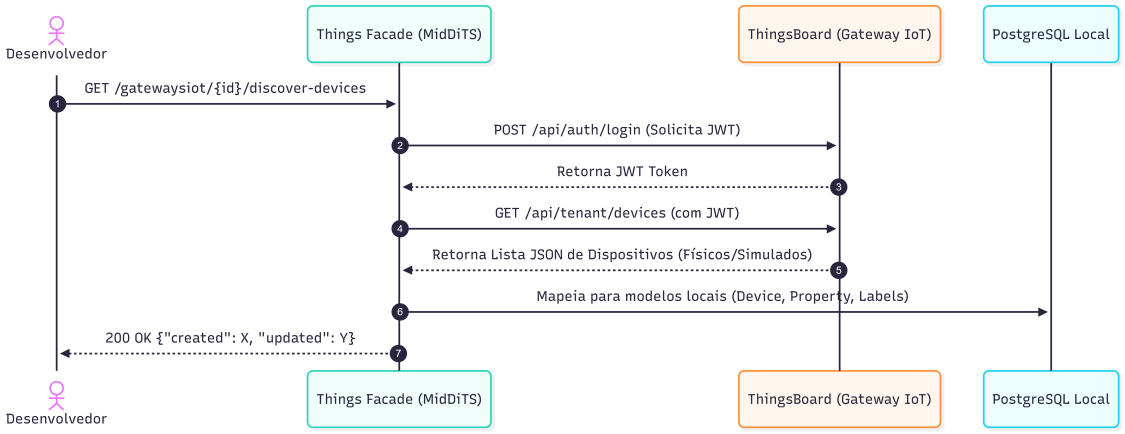


Figura 5.11. Fluxo de descoberta de dispositivos a partir do *gateway* IoT

os modelos previamente definidos em DTDL são registrados no *middleware*, permitindo que o sistema compreenda as estruturas, propriedades, telemetrias e relacionamentos das entidades.

Além do registro dos modelos, essa etapa envolve a organização das propriedades dos DTs e dos dispositivos físicos em um espaço comum de representação, no qual cada propriedade virtual pode ser potencialmente associada a uma propriedade física correspondente. Para isso, o sistema organiza os metadados provenientes do *gateway* IoT e os relaciona com as definições estruturais dos modelos digitais, estabelecendo o conjunto de candidatos que será utilizado no processo de *binding*.

Por fim, o sistema é conectado a um ambiente de execução, que pode ser composto por dispositivos reais ou por um simulador IoT. No contexto deste trabalho, utilizamos um simulador IoT para permitir a execução de experimentos de forma controlada, sem dependência de infraestrutura física.

Ao final desta fase, o sistema encontra-se plenamente configurado, com modelos registrados, dispositivos descobertos e ambiente preparado. O MidDiTS está, portanto, pronto para a materialização das instâncias e o estabelecimento dos vínculos com o mundo físico, que ocorrem na fase de desenvolvimento.

5.5.3. Desenvolvimento – Instanciação, Relacionamento e *Binding* dos DTs

Com o ambiente configurado e os modelos previamente definidos, a fase de desenvolvimento no MidDiTS corresponde à materialização dos DTs como entidades operacionais. É nesse momento que as abstrações definidas na fase de projeto tornam-se instâncias concretas, interconectadas e vinculadas ao mundo físico.

Criação de instâncias de DTs. A partir dos modelos definidos em DTDL, o desenvolvedor cria as instâncias de DTs, representando a transição do modelo abstrato para uma entidade executável dentro do sistema. O MidDiTS utiliza o banco de dados baseado em grafos Neo4j para armazenar essas instâncias, o que permite representar como os diferentes DTs se conectam entre si. Dessa forma, em vez de tratar cada elemento de forma isolada, o sistema passa a enxergar casas, cômodos e dispositivos como parte de

uma rede interligada, facilitando a navegação, a análise contextual e a realização de consultas baseadas nessas conexões. Cada instância de DT torna-se então um nó com o rótulo `DIGITALTWIN` e cada uma de suas propriedades torna-se um nó satélite `TWINPROPERTY`, ligado ao DT pela aresta `HAS_PROPERTY`.

Para facilitar o desenvolvimento de sistemas com muitas instâncias de DTs (como um bairro inteiro, por exemplo), o MidDiTS expõe um *endpoint* de criação hierárquica. Em vez de criar os nós um a um, o desenvolvedor pode enviar um único objeto JSON representando a árvore do ambiente para realizar um cadastro em lote, conforme o Código 5.2.

```
{
  "House 1": {
      "Room 1": {
          "LightBulb 1": {},
          "Air conditioner 1": {}
      },
      "Garden 1": {
          "Irrigation 1": {}
      }
  }
}
```

Código 5.2. Especificação em JSON de uma casa de forma hierárquica

Estabelecendo relacionamentos no grafo. Sistemas de DTs não são apenas listas de dispositivos, mas sim redes de entidades interconectadas. Embora seja possível representar parte dessa organização de forma hierárquica (por exemplo, casa → cômodo → dispositivo), essa abordagem não é suficiente para capturar todas as relações existentes no sistema. Isso ocorre porque, em cenários reais, as conexões entre entidades não se limitam a estruturas em árvore. Um mesmo tipo de dispositivo, como uma lâmpada, pode estar associado a diferentes contextos (por exemplo, a um cômodo específico ou a uma área funcional como uma cozinha) dependendo da modelagem adotada. Dessa forma, o desenvolvedor pode estabelecer explicitamente os relacionamentos entre os DTs, conforme as regras definidas nos modelos DTDL. No Neo4j, o MidDiTS materializa essas conexões por meio da criação de arestas entre os nós, permitindo representar o sistema como uma rede flexível, navegável e semanticamente rica de entidades interconectadas. A Figura 5.12 mostra a representação de um sistema completo de uma casa e seus relacionamentos.

***Binding* entre DTs e dispositivos físicos.** O passo fundamental desta fase é o *binding*, no qual propriedades dos DTs são associadas a dispositivos físicos registrados no sistema. A partir dessa conexão, o MidDiTS se encarrega de sincronizar automaticamente as contrapartes físicas e digitais. O MidDiTS suporta duas abordagens para o estabelecimento de *binding*:

1) *Binding* explícito (manual): O desenvolvedor associa diretamente um dispositivo físico a uma propriedade do DT por meio de chamadas à API do *middleware*.
2) *Binding* semântico: O MidDiTS utiliza técnicas de processamento de linguagem natural para inferir automaticamente correspondências entre propriedades dos DTs e dispositivos físicos com base na similaridade entre seus contextos.

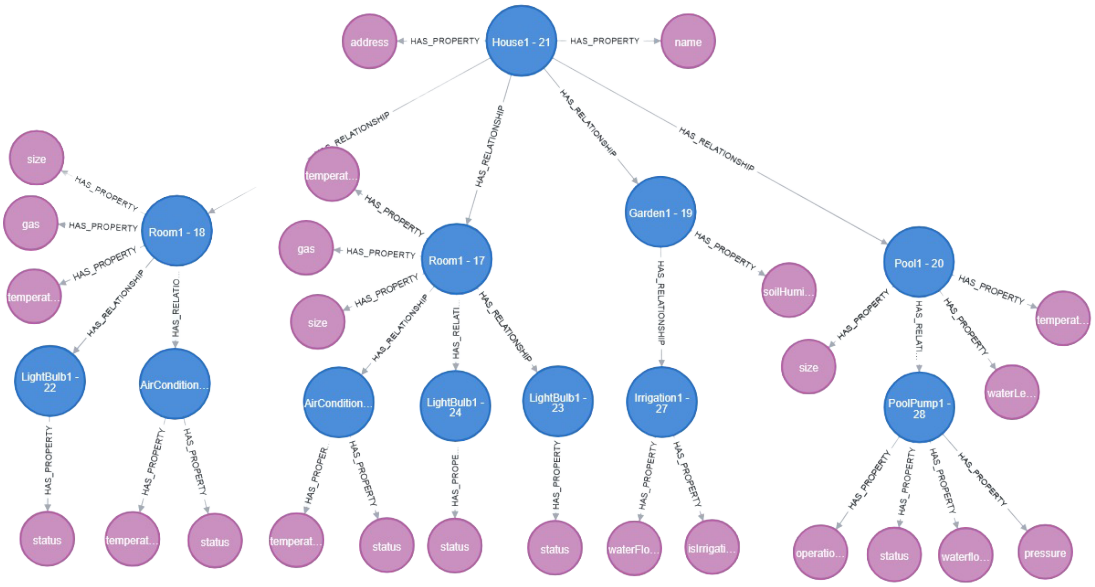


Figura 5.12. Representação de uma casa, sua hierarquia e relacionamentos no Neo4j

O MidDiTS considera o conjunto de metadados provenientes do *gateway* IoT, incluindo nomes, rótulos, atributos, descrições e chaves de telemetria. Em cenários com grande número de dispositivos, o processo de mapeamento individual dos dispositivos físicos às propriedades dos DTs torna-se inviável. Dessa forma, o *binding* semântico é realizado por meio de um modelo de *embeddings* de sentenças [Reimers and Gurevych 2019] (como o *all-MiniLM-L6-v2*[19]) para transformar tanto o contexto virtual quanto os metadados dos dispositivos físicos em representações vetoriais comparáveis. A similaridade entre esses elementos é calculada utilizando métricas como a similaridade de cosseno [Manning et al. 2008], permitindo identificar automaticamente correspondências mesmo quando há inconsistências ou variações na nomenclatura dos dispositivos.

Além da similaridade semântica capturada pelo modelo de *embeddings*, o MidDiTS implementa um *score* híbrido de correspondência. Esse *score* combina a proximidade semântica entre os textos descritivos com evidências adicionais extraídas dos nomes das instâncias, dos dispositivos e de formas canônicas das propriedades comparadas, reduzindo diferenças causadas por variações de escrita, prefixos, sufixos ou convenções distintas de nomeação. Esse mecanismo torna-se particularmente relevante em cenários com múltiplas entidades semelhantes. Por exemplo, em uma residência com *Room1* e *Room2*, ambas contendo um dispositivo chamado *AirConditioner1*, apenas a similaridade textual pode não ser suficiente para distinguir corretamente os vínculos. Ao considerar também o contexto da instância, o MidDiTS consegue associar a propriedade de temperatura de *Room1* ao *AirConditioner1* pertencente ao mesmo cômodo, evitando associações com dispositivos de outros ambientes. Com isso, o processo de sugestão de *binding* torna-se menos dependente apenas da similaridade textual e passa a considerar também sinais estruturais e contextuais presentes nos metadados.

[19]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Com base nesse *score* híbrido, o MidDiTS gera um conjunto de candidatos a *binding*, representando possíveis associações entre propriedades de DTs e dispositivos físicos. Esses candidatos são então ordenados conforme seu grau de correspondência e o *binding* pode ser estabelecido considerando um limiar mínimo de aceitação configurável pelo desenvolvedor. Esse limiar define o nível de confiança necessário para a associação e assume valores no intervalo $[0, 1]$, permitindo ajustar o nível de restrição das sugestões: valores mais altos retornam apenas associações de maior confiança, enquanto valores mais baixos ampliam o conjunto de candidatos considerados.

A Figura 5.13 ilustra o processo de estabelecimento de *binding* semântico, destacando a correspondência entre o contexto virtual e os metadados dos dispositivos físicos. Para operacionalizar esse processo, o MidDiTS disponibiliza APIs de *preview* e de efetivação de *binding*. A API de *preview* retorna uma lista ordenada de candidatos, permitindo ao desenvolvedor avaliar as sugestões antes de sua aplicação. Adicionalmente, o sistema permite restringir a análise a propriedades não vinculadas, considerar apenas propriedades causais, limitar o número de resultados e controlar o reúso de propriedades físicas.

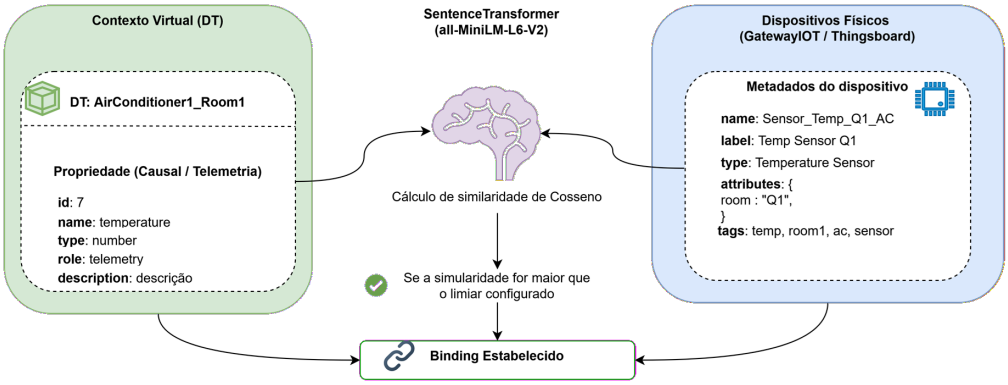


Figura 5.13. Representação do processo de *binding* semântico entre propriedades de DTs e dispositivos físicos.

Uma vez estabelecidos os *bindings*, o sistema passa a operar de forma contínua, com sincronização entre os estados físico e virtual. Nesse ponto, os DTs deixam de ser apenas representações informacionais e passam a atuar como entidades ciberfísicas, capazes de refletir e influenciar o comportamento do ambiente real, caracterizando o início da fase de operação.

5.5.4. Operação

Com a conclusão da fase de desenvolvimento, os DTs passam a operar como entidades ativas, mantendo uma sincronização contínua com suas contrapartes físicas. Na fase de operação, o foco desloca-se da construção estrutural para a manutenção da consistência, atualidade e fidelidade dos estados representados.

A operação no MidDiTS é sustentada por um fluxo bidirecional de dados, composto por dois processos principais: a propagação de telemetria do mundo físico para o digital (*sensor-to-middleware* – S2M) e a execução de comandos do modelo digital sobre os dispositivos físicos (*middleware-to-sensor* – M2S). Esse comportamento caracteriza

um ciclo contínuo de monitoramento, decisão e atuação, no qual aplicações externas observam o estado do ambiente físico, processam informações e influenciam diretamente esse ambiente por meio do DT, utilizando as APIs expostas pelo *middleware*. Internamente, essas operações são processadas por diferentes camadas do MidDiTS. A camada *Orchestration Layer* é responsável por interpretar as requisições sobre os DTs, enquanto a *Things Facade* gerencia a comunicação com o *gateway* IoT e a execução dos comandos nos dispositivos físicos.

Fluxo bidirecional. No fluxo S2M, dispositivos físicos enviam dados de telemetria, geralmente por meio de um *gateway* IoT (como a *ThingsBoard*). Esses dados são periodicamente capturados pelo MidDiTS, que atualiza as propriedades correspondentes nos DTs. Quando associadas a propriedades causais, essas atualizações não são meramente informativas, mas representam mudanças de estado que devem ser refletidas de forma consistente no modelo digital. Esse mecanismo garante que o DT permaneça alinhado com o estado atual do ambiente físico.

No sentido inverso, o fluxo M2S permite que alterações realizadas no modelo digital sejam propagadas para os dispositivos físicos por meio de comandos RPC. Por exemplo, ao alterar o estado de uma lâmpada no DT, o MidDiTS emite automaticamente uma requisição para o dispositivo correspondente, garantindo que a ação seja efetivamente executada no mundo físico. Esse comportamento é diretamente suportado pelas propriedades causais definidas na fase de modelagem.

Interface de operação via APIs. Durante a fase de operação, a interação com os DTs ocorre principalmente por meio das APIs expostas pelo MidDiTS. Essas APIs abstraem a complexidade da comunicação com os dispositivos físicos e permitem que aplicações externas manipulem os estados dos DTs de forma padronizada. De forma geral, o MidDiTS disponibiliza operações para: (i) consulta de estado de instâncias de DTs; (ii) atualização de propriedades, incluindo propriedades causais; (iii) execução de comandos associados a atuadores físicos; e (iv) monitoramento de eventos e telemetria.

Diferentemente de sistemas REST tradicionais, onde operações afetam apenas representações de dados, no contexto de DTs essas interações podem desencadear efeitos físicos no ambiente monitorado. Um exemplo de interação via requisições HTTP para consulta e atualização de propriedades de um DT é ilustrado nos Códigos 5.3 e 5.4, incluindo os respectivos *payloads* e respostas retornadas pelo *middleware*. Essas requisições seguem o padrão RESTful adotado pelo MidDiTS, permitindo que aplicações externas consultem estados e atuem sobre dispositivos físicos de forma transparente.

O Código 5.3 ilustra o consumo da API do MidDiTS para consulta de uma propriedade do DT e o respectivo reflexo no dispositivo físico. Quando essa propriedade é definida como causal, também é possível, por meio da API, manipular o valor de uma propriedade de um DT.

```
1 curl -X GET \
2 http://[host]/api/orchestrator/systems/1/instances/24/properties/7/
      value/ \
3 -H "Content-Type: application/json"
4 Resposta:
5 {
```

```
6     "value": "true"
7   }
```

Código 5.3. Consulta ao estado de uma propriedade do DT

O Código 5.4 mostra um exemplo de uma requisição HTTP PUT no MidDiTS alterando o status de uma propriedade de um DT e, consequentemente, do dispositivo físico.

```
1 curl -X PUT \
2 http://[host]/api/orchestrator/systems/1/instances/24/properties/7/
      value/ \
3 -H "Content-Type: application/json" \
4 -d '{"value": false}'
```

Código 5.4. Atualização de propriedade causal

A execução dessa requisição desencadeia uma sequência de interações internas no *middleware* e no *gateway* IoT, que resultam na atuação sobre o dispositivo físico. A Figura 5.14 ilustra esse fluxo de forma detalhada.

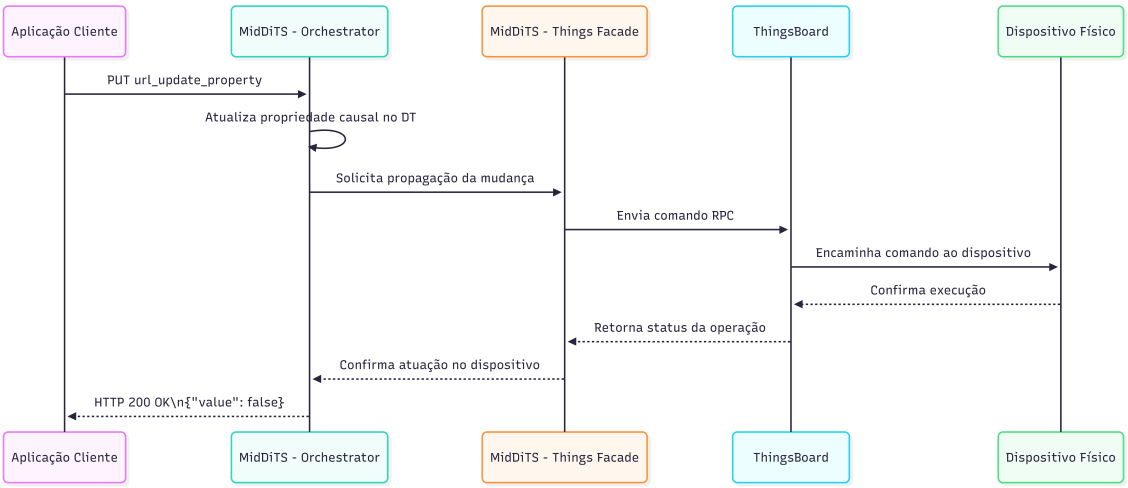


Figura 5.14. Representação do fluxo de atuação do modelo digital sobre o dispositivo físico (M2S)

Observa-se que não é necessário que o desenvolvedor conheça os mecanismos específicos de comunicação com o dispositivo físico, uma vez que o MidDiTS abstrai essa complexidade e gerencia automaticamente a propagação das alterações. Dessa forma, a API do MidDiTS não manipula apenas representações de dados, mas influencia diretamente o estado do mundo físico quando associada a propriedades causais.

O MidDiTS atua como elemento central na orquestração desses fluxos, sendo responsável por: (i) monitorar continuamente as atualizações de telemetria; (ii) identificar mudanças relevantes nas propriedades causais; (iii) garantir a consistência entre os estados físico e digital; e (iv) executar comandos e validar suas respostas. Essa operação contínua transforma o sistema em uma arquitetura ciberfísica, na qual decisões podem ser tomadas com base em um modelo digital atualizado em tempo quase real.

A fase de operação consolida o DT como um sistema dinâmico e responsivo. Diferentemente de uma representação estática, o DT passa a atuar como um agente ativo no ecossistema, capaz de refletir, analisar e influenciar o ambiente físico em tempo real. Nesse contexto, a operação não é apenas uma etapa do ciclo de vida, mas o elemento que sustenta a própria essência dos DTs: a manutenção de uma relação causal contínua e operacional entre o mundo físico e o digital, permitindo que decisões e ações sejam realizadas de forma consistente, confiável e responsiva. Esse comportamento operacional aproxima os DTs de sistemas ciberfísicos reativos, nos quais o modelo digital não apenas representa, mas participa ativamente da dinâmica do ambiente físico.

5.6. Conclusão

Este capítulo apresentou os principais conceitos, desafios e tecnologias relacionados a sistemas de DTs, bem como o uso de plataformas de *middleware* como estratégia para viabilizar sua construção de forma sistemática. Como discutimos ao longo do texto, o sucesso de um sistema de DT não reside apenas na coleta de dados, mas na capacidade de manter uma sincronia bidirecional e causal entre os domínios físico e digital.

Através deste minicurso, foi possível identificar a série de desafios técnicos e tecnológicos envolvidos na construção de um sistema de DTs, de modo a lidar com seus requisitos específicos. Observou-se ainda como um *middleware* atua como importante facilitador nesse contexto: o uso de camadas de abstração, como exemplificado pelo MidDiTS, é essencial para gerenciar a heterogeneidade e reduzir a complexidade do desenvolvimento. Nesse cenário, descrevemos e comparamos algumas soluções proprietárias e de código aberto, bem como outras propostas na literatura. A análise comparativa evidenciou que nenhuma dessas plataformas atende simultaneamente a todos os requisitos de um ecossistema de DT, motivando o desenvolvimento de soluções complementares.

Por fim, demonstramos o uso do MidDiTS como *middleware* para a construção de sistemas de DTs ao longo de quatro fases do ciclo de vida: projeto, configuração, desenvolvimento e operação. Na fase de projeto, apresentamos como construir modelos utilizando a linguagem DTDL, especificando elementos do ambiente do mundo real, dispositivos, propriedades de telemetria e causais e relacionamentos entre as entidades existentes. Na fase de configuração, foram apresentados o registro de tais modelos DTDL, a integração com o *gateway* IoT *ThingsBoard* e o uso de um simulador para validação independente de hardware físico. Na fase de desenvolvimento, foram abordadas a instanciação de DTs, o estabelecimento de relacionamentos entre entidades e o *binding* entre propriedades virtuais e dispositivos físicos. Na fase de operação, foram descritos os fluxos bidirecionais de sincronização e o papel central do MidDiTS na orquestração contínua dos estados físico e digital, transformando o sistema em uma arquitetura ciberfísica dinâmica e responsiva.

Em suma, o conteúdo deste capítulo demonstra que a adoção de um *middleware* especializado, orientado por um processo sistemático de desenvolvimento, é uma abordagem viável e eficaz para mitigar alguns dos desafios inerentes à construção de sistemas de DTs, promovendo interoperabilidade, reduzindo a complexidade de integração e habilitando a sincronização contínua entre os mundos físico e o digital ao longo de todo o ciclo de vida da entidade-alvo. Esperamos que este material sirva como guia para desenvolve-

dores e pesquisadores explorarem o potencial transformador dos DTs na próxima década de transformação digital.

Agradecimentos

Este trabalho possui apoio do Instituto Nacional de Ciência e Tecnologia em Redes de Comunicação e Internet das Coisas Inteligentes – INCT ICoNIoT (http://iconiot.ic.unicamp.br), financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, processo nº 405940/2022-0, e pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, processo nº 88887.954253/2024-00.

A elaboração deste capítulo contou com o apoio de ferramentas de Inteligência Artificial generativa para produzir algumas figuras e para a revisão do fluxo textual. Ressalta-se que todos os elementos gerados foram cuidadosamente revisados e validados pelos autores, que permanecem integralmente responsáveis pelo conteúdo e pela qualidade final deste capítulo, em plena conformidade com o Código de Conduta para Autores em Publicações da Sociedade Brasileira de Computação (SBC).

Referências

Almeida, A., Batista, T., Cavalcante, E., Delicato, F., Motta, R., and Vieira, M. (2023). Middleware for digital twins: A systematic mapping study. In *Proceedings of the 1st International Workshop on Middleware for Digital Twin*, pages 19–24, USA. ACM.

André, P., Azzi, F., and Cardin, O. (2020). Heterogeneous communication middleware for digital twin based cyber manufacturing systems. In Borangiu, T., Trentesaux, D., Leitão, P., Boggino, A. G., and Botti, V., editors, *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*, volume 853 of *Studies in Computational Intelligence*, pages 146–157. Springer International Publishing, Switzerland.

Bader, S., Barnstedt, E., Bedenbender, H., Berres, B., Billmann, M., and Ristin, M. (2022). Details of the Asset Administration Shell: Part 1 – The exchange of information between partners in the value chain of Industrie 4.0. Technical report, Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany.

Barbie, A., Hasselbring, W., and Hansen, M. (2024). Digital twin prototypes for supporting automated integration testing of smart farming applications. *Symmetry*, 16(2):221.

Bellavista, P., Bicocchi, N., Fogli, M., Giannelli, C., Mamei, M., and Picone, M. (2024). An entanglement-aware middleware for digital twins. *ACM Transactions on Internet of Things*, 5(4):25.

Bernstein, P. A. (1996). Middleware: A model for distributed system services. *Communications of the ACM*, 39(2):86–98.

Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2011). *Distributed systems: Concepts and design*. Addison-Wesley, USA, 5th edition.

Fuller, A., Fan, Z., Day, C., and Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971.

Iranshahi, K., Brun, J., Arnold, T., Sergi, T., and Müller, U. C. (2025). Digital twins: Recent advances and future directions in engineering fields. *Intelligent Systems with Applications*, 26.

ISO (2023). ISO/IEC 30173:2023: *Digital twin – Concepts and terminology*. https://www.iso.org/standard/81442.html.

Kritzinger, W., Karner, M., Traar, G., Henjes, J., and Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022.

Kuts, V., Modoni, G., Otto, T., Sacco, M., Tähemaa, T., Bondarenko, Y., and Wang, R. (2019). Synchronizing physical factory and its digital twin through an IIoT middleware: a case study. *Proceedings of the Estonian Academy of Sciences*, 68(4):364–370.

Lin, C.-T. and Lu, H.-J. (2024). An intelligent product-driven manufacturing system using data distribution service. *IEEE Access*, 12:16447–16461.

Liu, M., Fang, S., Dong, H., and Xu, C. (2021). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58:346–361.

Mahdi, M. M., Bajestani, M. S., Noh, S. D., and Kim, D. B. (2025). Digital twin-based architecture for wire arc additive manufacturing using OPC UA. *Robotics and Computer-Integrated Manufacturing*, 94.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.

Minerva, R., Lee, G. M., and Crespi, N. (2020). Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, 108(10):1785–1824.

Modoni, G. E., Veniero, M., Trombetta, A., Sacco, M., and Clemente, S. (2018). Semantic based events signaling for AAL systems. *Journal of Ambient Intelligence and Humanized Computing*, 9(5):1311–1325.

Pereira, L. S., Almeida, A., Delicato, F. C., Cavalcante, E., Batista, T., Motta, R., and Vieira, M. (2024). Leveraging middleware to support digital twin system development. In *Proceedings of the 2024 IEEE Smart World Congress*, pages 2081–2088, USA. IEEE.

Pylianidis, C., Osinga, S., and Athanasiadis, I. N. (2021). Introducing digital twins to agriculture. *Computers and Electronics in Agriculture*, 184.

Redelinghuys, A. J. H., Basson, A. H., and Kruger, K. (2020). A six-layer architecture for the digital twin: a manufacturing case study implementation. *Journal of Intelligent Manufacturing*, 31(6):1383–1402.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3980–3990, USA. Association for Computational Linguistics.

Schantz, R. E. and Schmidt, D. C. (2002). Research advances in middleware for distributed systems: State of the art. In Chapin, L., editor, *Communication Systems*, vol-

ume 92 of *IFIP – The International Federation for Information Processing*, pages 1–36. Springer New York, USA.

Shahat, E., Hyun, C. T., and Yeom, C. (2021). City digital twin potentials: A review and research agenda. *Sustainability*, 13(6).

Tao, F., Zhang, H., Liu, A., and Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415.

Tao, F. and Zhang, M. (2017). Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing. *IEEE Access*, 5:20418–20427.

van Schalkwyk, P., Whiteley, S., and Goldman, M. (2025). Digital Twin Capabilities Periodic Table - Version 1.2. https://www.digitaltwinconsortium.org/initiatives/capabilities-periodic-table/.

Yun, S., Park, J.-H., and Kim, W.-T. (2017). Data-centric middleware based digital twin platform for dependable cyber-physical systems. In *Proceedings of the 2017 Ninth International Conference on Ubiquitous and Future Networks*, pages 922–926, USA. IEEE.

Zhang, X., Lim, S., Lee, C., Song, W. S., Kim, Y. C., Yu, M., Hong, S. H., Yoo, N. H., and Wei, M. (2023). Integration of 5G and OPC UA for smart manufacturing of the future. In *Proceedings of the 2023 IEEE/SICE International Symposium on System Integration*, USA. IEEE.