FLeer2FLeer: Uma Ferramenta Web Baseada em Arquitetura Par-a-Par para Orquestração do Aprendizado Federado

João e Silva Costa[1], Guilherme A. Thomaz[1], Ronaldo A. Ferreira[2] e Miguel Elias M. Campista[1*]

[1]GTA-PEE/COPPE-DEL/Poli, Universidade Federal do Rio de Janeiro (UFRJ)
[2]Facom, Universidade Federal de Mato Grosso do Sul (UFMS)

{joaocosta,guiaraujo,miguel}@gta.ufrj.br,raf@facom.ufms.br

Abstract. *This work presents FLeer2FLeer (F2F), a web-based tool for federated learning orchestration based on a P2P architecture. F2F introduces an Indexer responsible for the discovery, advertisement, and admission of clients into multiple federations, without interfering with the training process. Built on the Flower framework, the system employs hook-based strategies for non-intrusive metric collection. The tool provides real-time monitoring of multiple training processes, network expansion through the addition of servers, and failure reporting. Experimental results show that the Indexer imposes minimal overhead on the network, while the training load is distributed across different servers, promoting greater scalability.*

Resumo. *Este trabalho apresenta o FLeer2FLeer (F2F), uma ferramenta web para orquestração de aprendizado federado baseada em uma arquitetura P2P. O F2F introduz um Indexador responsável pela descoberta, anúncio e ingresso de clientes nas múltiplas federações, sem interferir no processo de treinamento. Implementado sobre o framework Flower, o sistema utiliza estratégias baseadas em hooks para coleta não intrusiva de métricas. A ferramenta oferece monitoramento em tempo real de múltiplos treinamentos, expansão da rede pela adição de servidores e reporte de falhas. Os resultados experimentais demonstram que o Indexador impõe sobrecarga mínima à rede, enquanto a carga de treinamento é distribuída entre diferentes servidores promovendo maior escalabilidade.*

1. Introdução

O Aprendizado Federado (*Federated Learning* – FL) é uma técnica atualmente muito popular, na qual modelos de aprendizado de máquina são treinados por múltiplos clientes distribuídos. Os parâmetros resultantes de cada treinamento são enviados para um servidor agregador, que é responsável por combinar as contribuições individuais em um único modelo. Esse procedimento é repetido até que o modelo convirja ou algum critério de parada seja alcançado. Essa abordagem de treinamento distribuído preserva a privacidade dos usuários, pois apenas os parâmetros do modelo são compartilhados com o servidor central, em vez dos dados brutos gerados pelos clientes. Dessa forma,

*Este trabalho foi realizado com apoio do CNPq (310234/2025-5, 407304/2025-8, 408255/2023-4 e 405940/2022-0); da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Código de Financiamento 001 e 88887.954253/2024-00; da FAPERJ (E-26/200.438/2026 e E-26/210.778/2025) e da FAPESP (2023/00673-7 e 2023/00811-0).

o aprendizado federado torna-se interessante para situações em que o treinamento exige informações sensíveis dos usuários, como dados provenientes de sistemas de Internet das Coisas (IoT), médicos ou produzidos por dispositivos móveis [Bochie et al. 2021].

Apesar de seus benefícios, a implementação de sistemas de FL em larga escala enfrenta desafios, especialmente no que tange à escalabilidade e à orquestração dos diferentes treinamentos. A falta de um serviço de treinamento federado e a potencial dependência de um único servidor central podem criar desestímulos à adoção do FL ou gargalos computacionais. Uma solução promissora, abordada por Ching *et al.*, para mitigar esses problemas é a adoção de arquiteturas par-a-par (*Peer-to-Peer* – P2P), nas quais os clientes interessados podem se organizar em múltiplas federações para treinar modelos ou disponibilizá-los para treinamento [Ching et al. 2024]. Um sistema P2P que orquestre o treinamento federado pode representar um ponto de referência para adoção em larga escala, além de naturalmente distribuir a carga de trabalho, característica herdada do P2P. Cada participante pode atuar tanto como cliente quanto como servidor, aumentando a resiliência e a capacidade de processamento, caso múltiplas federações existam. Entretanto, o trabalho de Ching *et al.* não foca na usabilidade, tornando a participação em uma tarefa de FL um processo mais complicado do que a participação em uma rede P2P típica. Assim, a escassez de ferramentas que facilitem a busca, o ingresso e o monitoramento em tarefas de FL ainda é uma lacuna que limita a sua ampla adoção.

Neste contexto, este trabalho propõe o desenvolvimento de uma ferramenta web inspirada na arquitetura P2P, chamada de FLeer2FLeer (F2F), para treinamento de modelos federados. O F2F propõe um servidor operando como Indexador para descoberta e gerenciamento de federações. Diferente das abordagens focadas em mudanças complexas no algoritmo, como observado em [Manhaes et al. 2025], esta ferramenta prioriza a experiência do usuário e a governança de múltiplos treinamentos de modelos simultâneos. O sistema desenvolvido oferece uma interface visual para anúncio, consulta e ingresso em uma das federações disponíveis; candidatura de novos servidores; monitoramento em tempo real de múltiplos treinamentos; e um mecanismo de reporte de falhas, visando facilitar a adesão de novos participantes e a manutenção da integridade das federações. A implementação é baseada no Flower [Beutel et al. 2022], um arcabouço de FL amplamente utilizado, para facilitar sua adoção por desenvolvedores já familiarizados.

Este trabalho está organizado da seguinte forma: a Seção 2 explica o funcionamento do Flower para o treinamento federado. Em seguida, a Seção 3 detalha a arquitetura do F2F, enquanto a Seção 4 valida a ferramenta por meio da apresentação de sua interface e de um experimento que avalia a distribuição de tráfego de rede. Por fim, a Seção 5 conclui o estudo e aponta direções para trabalhos futuros.

2. Aprendizado Federado com o Flower

O treinamento do aprendizado federado (FL) difere do centralizado tradicional. Para viabilizar essa arquitetura mais complexa, o Flower apresenta-se como uma opção de arcabouço unificado, capaz de abstrair a complexidade da infraestrutura do FL e ainda permitir a compatibilidade com bibliotecas de aprendizado de máquina populares, como o PyTorch e o TensorFlow [Beutel et al. 2022]. O funcionamento do Flower, baseia-se na interação de duas entidades principais: cliente (ClientApp), abstração dos clientes do FL, responsáveis por manter os dados locais e executar o treinamento dos modelos; e servidor

(ServerApp), abstração do servidor agregador do FL, responsável pela orquestração do treinamento, seleção de clientes por rodada e agregação dos parâmetros recebidos.

A lógica que rege a interação entre as duas entidades (cliente e servidor) é a estratégia (Strategy), que é responsável por definir a política de seleção de clientes, a configuração do treinamento local e o método de agregação dos modelos. O Flower oferece flexibilidade para customizar o processo de aprendizado no servidor, permitindo quatro abordagens principais: o uso de estratégias pré-existentes, a personalização via funções de *callback* no método de inicialização, a sobrescrita de métodos específicos de uma estratégia ou a implementação completa de um novo algoritmo do zero. Dentre as opções existentes, a estratégia padrão adotada, denominada FedAvg (*Federated Averaging*), apresenta dois estágios, treinamento *(fit)* e validação *(evaluate)*, cada um com três etapas que se repetem a cada rodada:

Etapa 1 (Configuração e Seleção): Esta etapa é comum a ambos os estágios (*fit* e *evaluate*). Nela, o servidor define as instruções da rodada e seleciona uma fração dos clientes disponíveis (amostragem). Em seguida, envia os parâmetros atuais do modelo global e as configurações opcionais de treinamento ou validação (ex., taxa de aprendizado e número de épocas locais customizados) para os nós selecionados.

Etapa 2 (Processamento Local): Para o estágio de *fit* cada cliente selecionado treina o modelo recebido utilizando seus próprios dados locais. Ao final do processo, o cliente retorna ao servidor apenas os parâmetros atualizados do modelo (pesos) e métricas, como o número de amostras utilizadas. Já para o estágio de *evaluate* cada nó valida o novo modelo global, gerado após o treinamento, em sua própria base de dados, retornando métricas como perda e acurácia para o servidor.

Etapa 3 (Agregação): Nesta etapa, os estágios diferem no conteúdo a ser agregado pelo servidor. No *fit*, o servidor coleta as contribuições de todos os clientes e executa uma média ponderada dos parâmetros, usando como peso, a quantidade de amostras de cada cliente para gerar um novo modelo global. No *evaluate*, o servidor segue o mesmo processo, mas com as métricas para determinar uma medida de desempenho global.

Os estágios *fit* e *evaluate* são executados de forma alternada. Assim, após a execução da Etapa 3 do *fit*, por exemplo, o sistema retorna à Etapa 1, agora para o estágio de *evaluate*. O processo é repetido enquanto um critério de parada não for alcançado.

Por fim, a infraestrutura de comunicação do Flower abstrai a complexidade da rede, utilizando por padrão o protocolo gRPC, que usa como base o HTTP2, para a troca eficiente de mensagens de controle e *payloads* de modelos. Para permitir um controle sobre o ciclo de vida do treinamento, a ferramenta expõe pontos de interceptação na estratégia, conhecidos como *hooks* de configuração. Funções como a `on_fit_config_fn` e a `on_evaluate_config_fn` permitem que o servidor construa e envie dicionários de configuração dinâmicos para os clientes imediatamente antes do início das etapas de treinamento (*fit*) e validação (*evaluate*), respectivamente. Esses mecanismos viabilizam o envio de informações variadas a cada rodada, como o ajuste de hiperparâmetros ou solicitações de métricas específicas, sem a necessidade de alterar a lógica interna dos clientes. Um desafio em aberto, porém, é a ausência de um serviço de descoberta de federações, além da ausência da orquestração de múltiplas tarefas distintas de aprendizado federado ocorrendo concomitantemente. Ou o cliente participa involuntariamente

de um procedimento de treinamento federado, como em uma aplicação que já possui o *ClientApp* executando no *backend*, ou precisa conhecer de antemão o servidor federado.

3. FLeer2FLeer

Esta seção detalha a arquitetura (Figura 1) e a implementação da ferramenta proposta, o FLeer2FLeer (F2F). Inspirada na troca de arquivos BitTorrent, tal ferramenta foi projetada para atuar como uma plataforma que, por meio da intermediação de federações, auxilia e fomenta o ingresso de novos participantes nesse ambiente de aprendizado federado. Sua arquitetura baseia-se na presença de um Indexador que, assim como em um sistema P2P para troca de arquivos, responde a consultas sobre fontes de serviços de interesse, na forma de tarefas de aprendizado federado. Potenciais clientes interessados podem ingressar em uma federação e colaborarem com seus dados locais, auxiliando na tarefa de treinamento com manutenção de privacidade. Assim, a ferramenta oferece as seguintes funcionalidades:

- Anúncio, consulta e ingresso em uma das federações disponíveis para participação no treinamento ou aquisição do modelo previamente treinado;
- Monitoramento em tempo real dos parâmetros de múltiplos treinamentos e exibição das métricas coletadas por meio de uma interface web;
- Serviço de expansão da rede F2F, a partir da candidatura de novos servidores;
- Serviço de reporte de falhas na federação.

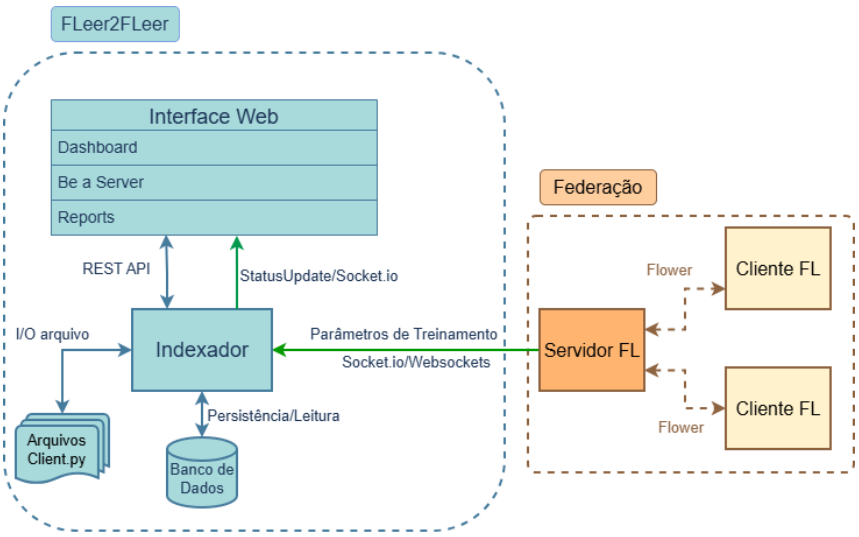


Figura 1. Representação da arquitetura da ferramenta.

3.1. Anúncio, Consulta e Ingresso no Sistema

A intermediação de federações constitui a operação central do F2F, sendo o pilar que viabiliza a maioria de suas funcionalidades. Tal processo é centralizado no Indexador, que não interfere em nenhuma das etapas do Flower descritas na Seção 2, preservando a privacidade e a autonomia das federações. A seguir, o fluxo para anúncio e consulta às federações existentes, com posterior ingresso para participação em uma delas, é detalhado. Todos os procedimentos são executados mediante a intermediação do Indexador.

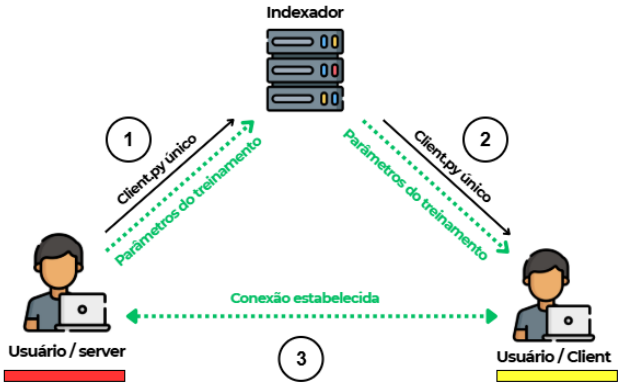


Figura 2. Exemplo de anúncio, consulta e ingresso em uma federação, sendo esses procedimentos intermediados pelo Indexador.

Etapa 1 (Anúncio de disponibilidade): o servidor do FL anuncia sua disponibilidade, tornando-se visível aos clientes por meio de uma interface web (*dashboard* web) do Indexador. O servidor do FL pode ser um usuário comum que quer inicializar um procedimento de treinamento de um modelo em sua máquina local, um servidor corporativo na nuvem ou o próprio Indexador, caso este acumule funções. Ao anunciar um servidor de FL, os arquivos necessários para a participação dos clientes (ex., o `Client.py`) são disponibili zados para o Indexador, que aguarda a consulta e a posterior solicitação de ingresso de clientes do FL interessados.

Etapa 2 (Consulta): Antes de ingressar em uma federação, o cliente do FL consulta o Indexador para conhecer os servidores do FL disponíveis. A interface web oferece métricas adicionais de treinamento que auxiliam a tomada de decisão do cliente do FL.

Etapa 3 (Ingresso na federação): Ao solicitar o ingresso em uma federação ao Indexador, o cliente recebe os arquivos necessários (ex., o `Client.py`), disponibilizados pelo Indexador na Etapa 1. Após receber os arquivos, o cliente estabelece conexão diretamente com o servidor do FL, como feito no Flower. Durante o treinamento, o Indexador coleta parâmetros para manutenção de métricas que auxiliem novos clientes na decisão sobre o ingresso em uma federação, como ilustrado na Figura 2.

Note que caso não haja um servidor do FL disponível para atender um cliente, este não executa a Etapa 3, podendo repetir a consulta (Etapa 2) posteriormente. Porém, para que haja um servidor de FL disponível, é necessário que a Etapa 1 seja realizada.

3.2. Monitoramento e Exibição de Métricas

A comunicação entre o servidor do FL e o Indexador foi implementada por meio de WebSockets, utilizando a biblioteca `Socket.IO`. Essa comunicação permite a coleta de métricas que são usadas posteriormente para atualização da interface web do Indexador, com as métricas de cada servidor do FL anunciado. Para a instrumentação das métricas, optou-se pela adaptação do código do servidor do FL, adicionando um pequeno bloco de código que explora os *hooks* globais de configuração (funções `on_fit_config_fn` e `on_evaluate_config_fn`). A escolha dessas funções justifica-se pela sua simplicidade e alto grau de generalidade, permitindo que atuem como *callbacks* não intrusivos que expõem o estado da federação a cada estágio

do ciclo de treinamento, conforme discutido na Seção 2. Essa abordagem assegura a autonomia dos servidores para definir suas próprias estratégias, viabilizando assim a oferta de uma ampla diversidade de tipos de servidores do FL e aplicações para novos clientes, sem impor restrições rígidas de implementação.

Após a coleta das métricas, cabe ao Indexador orquestrar a interface web por meio da qual os status de treinamento de cada federação são exibidos aos usuários. A manutenção do status ocorre a partir do recebimento de dados provenientes das funções de *callback* e da execução de tarefas em segundo plano. Essa rotina de orquestração da interface web do Indexador a partir da coleta de métricas é composta por três operações:

Lógica de Atualização do Banco de Dados: o Indexador analisa o *payload* recebido para identificar os campos associados ao servidor correspondente no banco de dados. Alguns desses campos são atualizados por incremento, enquanto outros, que representam médias, são recalculados com base nos valores previamente armazenados.

Monitoramento de Disponibilidade (*Heartbeat*): uma tarefa em segundo plano é executada periodicamente (a cada 5 minutos) para auditar a lista de servidores ativos. O algoritmo compara o *timestamp* da última comunicação de um servidor com um limiar de tolerância pré-definido. Caso um servidor cesse a comunicação além desse limite, seu estado é transicionado automaticamente para "offline", assegurando a confiabilidade da interface apresentada.

Atualização da interface web: simultaneamente ao processamento, os novos campos são difundidos para a interface web do Indexador via *WebSockets*. Isso garante a atualização em curto prazo de parâmetros na interface, como o progresso do treinamento e a carga de clientes conectados, fornecendo ao usuário as informações necessárias tanto para a seleção de um servidor quanto para o auxílio na conexão de novas federações.

3.3. Expansão da Rede F2F

Para garantir a escalabilidade descentralizada prevista na arquitetura, implementa-se um fluxo de admissão de novos servidores do FL. O serviço de candidatura, ou "*Be a Server*", permite que usuários do F2F assumam o papel de servidores, tornando-os visíveis ao Indexador. A ideia é validar os scripts que os servidores candidatos distribuem aos seus clientes. Note que o funcionamento de uma federação requer, além da definição do modelo a ser treinado, todo o software de execução do modo cliente do Flower, ou seja, do `Client.py`. Assim, a candidatura de um novo servidor inclui as seguintes etapas:

1. **Envio de parâmetros e do script `Client.py`:** o usuário que deseja registrar um servidor deve preencher um formulário estruturado fornecendo detalhes do modelo (categoria e aplicação) e realizar o *upload* do *script* de execução compatível (ex., `Client.py`), ao seu ServerApp e Strategy. Isso garante que todos os clientes que ingressem na federação possam se conectar ao servidor e iniciar o treinamento sem falhas.
2. **Validação:** o Indexador verifica a integridade dos arquivos recebidos antes de permitir a exibição em sua interface web.
3. **Registro:** após a validação, o novo servidor é registrado no banco de dados, tornando-se imediatamente visível na interface web e, consequentemente, disponível para consultas e solicitações de ingresso.

3.4. Reporte de Falhas

A confiabilidade de sistemas distribuídos é frequentemente comprometida pela presença de nós maliciosos ou instáveis. Para mitigar esse risco sem depender de uma moderação centralizada e manual, desenvolveu-se um mecanismo de reporte de falhas. Este mecanismo oferece uma interface que permite que usuários notifiquem anomalias na federação, como um servidor mal-intencionado. Para assegurar a utilidade dos dados reportados, o formulário exige a classificação do incidente da seguinte forma:

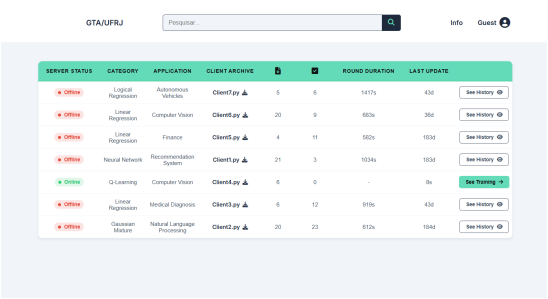
Identificação do alvo: o ID do servidor em que a falha foi detectada.

Tipificação técnica: categoria do erro (ex., *bug*, falha de conexão, nó malicioso).

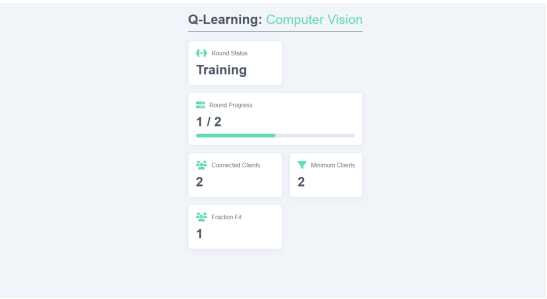
Essa abordagem promove uma governança comunitária, na qual a própria base de usuários participa da manutenção da integridade e da qualidade do ecossistema do F2F.

4. Validação da Ferramenta

A interface web do Indexador está ilustrada em *prints* da tela. A Figura 3a apresenta a interface web, contendo informações sobre os modelos já treinados ou em treinamento; o número de *downloads*, que indica o nível de adesão dos clientes; e a duração da rodada de treinamento em segundos, que indica o tamanho do modelo. Já a Figura 3b demonstra a tela usada para acompanhar o progresso do treinamento em servidores online, acessível pelo botão "*See Training*". Essas ilustrações permitem compreender o funcionamento prático da ferramenta e parte das funcionalidades descritas na Seção 3.



(a) Servidor disponível para treinamento federado.



(b) Acompanhamento do treinamento de uma federação.

Figura 3. Inferface web do Indexador F2F.

A ferramenta foi disponibilizada em um repositório no GitHub, acessível pelo link (`https://github.com/Jscosta7/FLeer2FLeer`). O repositório contém um arquivo `README.md` com os artefatos necessários para executar a ferramenta e os experimentos desta seção. Para a demonstração da ferramenta no SBRC será necessário um notebook com acesso a internet. O experimento foi planejado para avaliar a distribuição do tráfego de rede em duas frentes: no Indexador, observando se sua orquestração gera sobrecarga com o aumento do número de federações; e nos servidores, onde se espera um padrão de tráfego constante por federação. Utilizou-se o Docker para o isolamento das entidades do F2F e das federações. Na implementação do aprendizado federado, adotou-se a arquitetura MobileNetV2 (via TensorFlow/Keras), configurada para entradas de $32 \times 32 \times 3$ pixels e classificação de 10 categorias. Como conjunto de dados, utilizou-se o CIFAR-10, composto por 60.000 amostras rotuladas. A execução do experimento

variou de 1 a 5 o número de federações, sendo que cada federação é composta por um servidor e dois clientes. Cada barra nos gráficos corresponde à média do tráfego de rede obtida a partir de cinco repetições.

A Figura 4 mostra que o tráfego do Indexador é relativamente baixo, se mantendo próximo de 0,1% do tráfego total ("Total (FL)"). Esse resultado indica a escalabilidade do F2F com o aumento do número de federações. Além disso, a Figura 4 demonstra que a média de tráfego por servidor ("por Servidor") permanece constante em relação ao número de federações, confirmando que a arquitetura baseada em P2P tende a dividir a carga da rede entre os múltiplos servidores. O eixo Y da figura está em escala logarítmica para ilustrar o tráfego de rede do Indexador, que é três ordens de magnitude inferior.

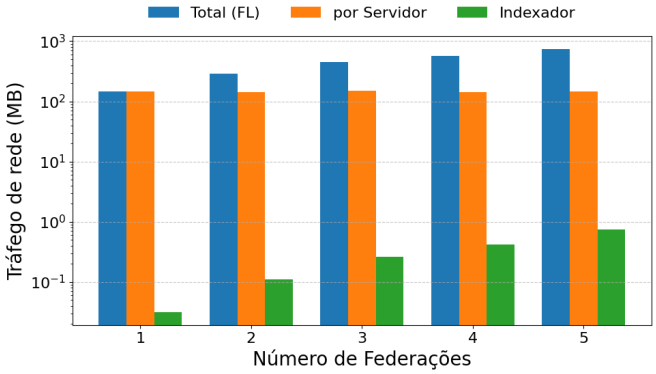


Figura 4. Tráfego de rede gerado pelo Indexador em comparação com o tráfego total de treinamento e com o tráfego de cada servidor de FL.

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou o FLeer2FLeer (F2F), uma ferramenta projetada para facilitar o treinamento de modelos federados em múltiplas federações e o ingresso de novos clientes participantes. O F2F oferece diversas funcionalidades sem comprometer a autonomia das federações, contribuindo para a diversidade de servidores na plataforma. Além disso, a avaliação experimental indica que a arquitetura é escalável, uma vez que a intermediação centralizada no Indexador provê visibilidade e orquestração aos usuários com um impacto negligenciável no tráfego total da rede. Como trabalhos futuros, reconhece-se a necessidade de avaliar o desempenho sob estresse, aferindo o consumo de CPU, memória RAM e a latência na conexão dos clientes, além de desenvolver mecanismos de segurança para proteger a distribuição do `Client.py`. Adicionalmente, pretende-se aprimorar o sistema de consultas, fornecendo métricas mais detalhadas sobre os modelos e o estado das federações, bem como otimizar as recomendações de ingresso com base na capacidade computacional dos nós.

Referências

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., and Lane, N. D. (2022). Flower: A Friendly Federated Learning Research Framework.

Bochie, K., Sammarco, M., Detyniecki, M., and Campista, M. E. M. (2021). Análise do aprendizado federado em redes móveis. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 71–84. SBC.

Ching, C.-W., Chen, X., Kim, T., Ji, B., Wang, Q., Da Silva, D., and Hu, L. (2024). Totoro: A scalable federated learning engine for the edge. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 182–199.

Manhaes, B. V., Ferreira, J. P. M., Thomaz, G. A., and Campista, M. E. M. (2025). Avaliaçao da privacidade diferencial aplicada ao aprendizado federado através de ataques de inversao de modelo. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pages 211–225. SBC.