Avaliação de Desempenho de Aplicações de Aprendizado Federado em Redes de Acesso Compartilhadas

Diogo M. Cunha, Marco A. Guerra, Oscar J. Ciceri, Nelson L. S. da Fonseca, and Carlos A. Astudillo

[1]Universidade Estadual de Campinas (UNICAMP), Campinas, Brazil
{d249418, m184714, o164786}@dac.unicamp.br,
nfonseca@ic.unicamp.br, castudillo@unicamp.br

Resumo. *O aprendizado federado (FL) permite treinar modelos de aprendizado de máquina (ML) por clientes distribuídos sem a necessidade de compartilhar seus dados locais com um servidor central (CS). Ao compartilhar somente os parâmetros locais dos modelos dos clientes, FL mitiga problemas de segurança e privacidade do treinamento tradicional de ML, reduzindo a exposição de dados sensíveis. No entanto, FL introduz uma nova classe de aplicações de rede, caracterizadas por troca de parâmetros frequentes e de grande tamanho, fazendo um uso significativo dos recursos de rede e computação, gerando desafios em situações com largura de banda e recursos de processamento limitados. Vários fatores estão diretamente relacionados à carga gerada na rede, como o tamanho do modelo e o número de clientes envolvidos. Embora a configuração desses parâmetros tenha como objetivo principal maximizar a precisão e a convergência do modelo, alcançar um equilíbrio entre a qualidade do modelo e os recursos de rede disponíveis é essencial. Este estudo analisa o impacto de diferentes aplicações de FL e seus parâmetros na rede de acesso. Para isso, foi desenvolvido um simulador de rede, e uma metodologia para gerar tráfego de FL a partir do simulador LEAF, que é um benchmark para aprendizado em ambientes federados. Os resultados da simulação mostram um aumento na latência do tráfego de FL à medida que o número de clientes aumenta ou o tamanho do batch diminui.*

Abstract. *Federated learning (FL) enables the training of machine learning (ML) models by distributed clients without sharing their local data with a central server (CS). By sharing only the local model parameters of the clients, FL addresses security and privacy challenges of traditional ML training, reducing sensitive data exposure. However, FL introduces a new class of network applications, characterized by frequent and large-size model parameter exchanges and significant network and computational resource utilization, leading to challenges in situations with limited bandwidth and processing resources. Several factors are directly related to the network traffic load generated, such as model size, the number of clients involved, and hyperparameter configuration. Although the configuration of these parameters primarily aims to maximize the model's accuracy and convergence, achieving a balance between model quality and available network resources is essential. This study analyzes the impact of relevant FL application factors under the same access network on the FL model and network performance. To this end, a homemade network simulator was developed, including a methodology for generating FL traffic and obtaining application-level performance from the LEAF framework, which is a benchmark for learning in federated settings. Simulation results show an increase in latency in FL traffic as the number of clients increases or the batch size decreases.*

1. Introdução

A Aprendizagem de Máquina (ML) tem atraído considerável atenção da academia e a industria devido à sua capacidade de resolver problemas complexos com baixo custo de

implementação. No contexto de redes Sexta Geração de Redes Móveis (6G), espera-se que ML otimize a alocação de recursos, preveja padrões de tráfego e melhore a experiência dos usuários. No entanto, os métodos tradicionais de ML exigem que para treinamento do modelo os clientes transmitam dados brutos para um Servidor Central (CS). Esse procedimento envolve requisitos relacionados à segurança e à privacidade dos dados, além de aumentar o trafego na rede. As regulamentações recentes sobre privacidade de dados impõem desafios ainda maiores na implementação dos paradigmas tradicionais de ML [Parliament 2016], ao exigir altos níveis de segurança para que os dados sejam transmitidos para servidores.

O Aprendizagem Federativo (FL) surgiu como uma abordagem para o treinamento de modelos de ML, preservando a privacidade [McMahan et al. 2017]. No FL, os clientes treinam modelos de ML localmente com seus dados, sem compartilhar os dados brutos com o CS, reduzindo o risco de vazamentos de dados. Em vez de transmitir os dados brutos, os clientes compartilham os parâmetros do modelo com o CS, o qual os agrega em um modelo global. Esse modelo global é então reenviado aos clientes para o próximo ciclo de treinamento. O processo iterativo continua até que o modelo atinja um precisão predefinida ou um número especificado de ciclos de treinamento seja completado.

Embora o FL melhore a privacidade, permitindo que os clientes treinem o modelo localmente, FL também introduz uma sobrecarga de trafego significativa na rede. Isso se deve ao grande tamanho dos modelos locais e à frequência das atualizações dos modelos entre os clientes e o CS. O tamanho dos modelos varia de kilobytes a gigabytes, dependendo da aplicação. Por exemplo, modelos simples como um classificador linear e o LeNet têm tamanhos de modelo, de 1,10 kB e 350 kB, respectivamente. Já arquiteturas baseadas em Transformer apresentam tamanhos de modelo significativamente maiores, por exemplo modelos Vision Transformer (ViT) alcançam 329,62 MB e modelos BERT ultrapassam 417,72 MB [Jin et al. 2023]. Como resultado, as aplicações de FL podem exigir grandes recursos de rede e de computação, o que representa um desafio para a implementação de modelos de FL em redes com largura de banda limitada ou dispositivos com poder computacional restrito.

No entanto, uma mesma aplicação FL pode gerar diferentes cargas de tráfego, dependendo da sua configuração. Hiperparâmetros como taxa de aprendizado, tamanho do lote (*i.e.,* batch size) e número de épocas, definem o tempo de treinamento e, portanto, a frequência com que os pacotes são enviados entre os clientes e o CS. A arquitetura e a complexidade da rede neural definem o número de parâmetros do modelo, o que determina o tamanho dos pacotes a serem enviados. Além disso, o número total de clientes impacta o volume geral de tráfego gerado durante o treinamento. Isso significa que é necessário configurar a aplicação de FL de acordo com os recursos computacionais e de rede disponíveis.

Aplicações com um alto consumo de recursos usualmente tem pacotes da ordem de centenas de MB, com alta frequência nas atualizações. Essas aplicações são apropriadas para cenários cross-silo, onde são poucos clientes federados implantados em computadores de alta capacidade, como data centers ou clusters, e conectados ao CS por acesso de banda larga. Por outro lado, aplicações que geram pacotes com tamanho no ordem dos kB com ciclos de treinamento longos são apropriados para cenários cross-device, onde os clientes possuem recursos computacionais limitados e um canal não confiável com largura de banda restrita.

Além disso, o tráfego das aplicações de FL compete com outros tipos de tráfego que coexistem nas redes de acesso, o que torna ainda mais desafiador o fornecimento de qualidade de serviço. Por exemplo, o tráfego FL pode competir com tráfegos sensíveis ao atraso, como multimídia ou VoIP. Como ilustrado na Figura 1, esses tráfegos provêm de clientes residenciais, empresariais ou até mesmo de operadoras de rede móvel, que utilizam a mesma rede de acesso em banda larga para enviar o tráfego de suas estações.

A seleção apropriada das configurações nas aplicações de FL é importante para

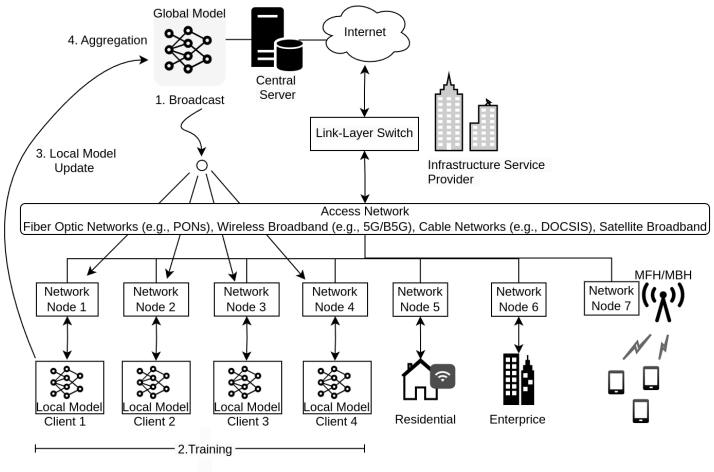


Figura 1. Arquitetura de Aprendizado Federado sobre Redes de Acesso em Banda Larga

gerar um equilíbrio entre o desempenho do modelo e os recursos de rede e computação disponíveis, especialmente ao implementá-las em ambientes com recursos limitados. Ao ajustar os parâmetros, é possível minimizar a sobrecarga de comunicação e as demandas computacionais, garantindo a viabilidade prática das aplicações FL em cenários com recursos restritos. Compreender a relação entre a configuração do modelo federado e o consumo de recursos da rede é essencial para equilibrar a precisão do modelo e o desempenho da rede, garantindo uma implementação escalável das aplicações FL em redes de comunicação existentes.

A caracterização dos tradeoffs entre o desempenho das aplicações e a comunicação tem sido recentemente abordada na literatura. Diversos estudos investigam o impacto de fatores como tráfego de fundo, perdas de pacotes e o tempo sincronização de clientes na carga da rede e no desempenho do modelo. Por exemplo, [Eriş et al. 2021] explora o impacto do tráfego de fundo na desempenho das aplicações FL, enquanto [Rodio et al. 2023] propõe o algoritmo para lidar com perdas de pacotes FL em redes sem fio. Além disso, [Tedeschini et al. 2023] apresenta um modelo de tráfego para otimizar a comunicação em redes sem fio com clientes heterogêneos. Essas estratégias visam melhorar a convergência dos modelos FL em redes com recursos limitados. No entanto, esses trabalhos ainda não avaliam o impacto das configurações das aplicações FL tanto no desempenho dos modelos quanto nas redes de comunicação. A falta de uma análise mais profunda sobre como as configurações podem afetar simultaneamente a eficácia do modelo e a eficiência da rede limita a aplicabilidade dessas abordagens em cenários reais, onde as condições de rede e os requisitos de desempenho podem variar consideravelmente.

Este trabalho analisa como o acesso a rede pode impactar diferentes configurações de aplicações de FL. Através dessa análise, é possível estabelecer uma relação entre os recursos computacionais e a demanda de largura de banda requeridos para diferentes configurações de aplicações FL. Dessa forma, busca-se contribuir para o desenvolvimento de estratégias mais eficientes que equilibram a qualidade do modelo com os recursos das redes de comunicação. Neste estudo, foi possível compreender como o ajuste de parâmetros das aplicações FL influencia o custo de comunicação. Para alcançar esse objetivo, propôs-se uma metodologia que integra o benchmark LEAF [Caldas et al. 2018], responsável pelo treinamento do modelo e pela geração do tráfego de FL, com um simulador do acesso à rede baseado em *traces* desenvolvido internamente. Essa integração visa criar uma plataforma robusta, capaz de modelar de forma realista os comportamentos e impactos do tráfego de FL em cenários complexos de comunicação, onde outros tráfegos estão competindo por recursos, como pode ser o caso do tráfego multimídia.

Foram utilizados dois conjuntos de dados FEMNIST [Cohen et al. 2017] e Shakespeare [Shakespeare 2014]. Em resumo, este artigo apresenta as seguintes contribuições:

- Desenvolvimento de um simulador de acesso baseado *traces* e propomos uma metodologia para a sua integração com emulador de aplicações de FL chamado LEAF. Este simulador, modela, de forma geral, a camada de enlace, para analisar o comportamento do acesso à rede quando submetida a tráfego FL, permitindo avaliar o desempenho do modelo e a influência do acesso no treinamento.
- Análise do impacto do acesso à rede no treinamento de aplicações FL.
- Recomendações para a configuração dos parâmetros das aplicações FL em cenários com recursos limitados, auxiliando os profissionais a equilibrar o desempenho do modelo e a utilização dos recursos de rede.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os principais trabalhos relacionados a esta pesquisa. A Seção 3 introduz a metodologia do trabalho, incluindo o modelo de sistema e os detalhes de nossa proposta. A Seção 4 aborda os parâmetros de simulação e a discussão dos resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e direções para trabalhos futuros

2. Trabalhos Correlatos

O estado da rede representa um fator importante no tempo de convergência e precisão do modelo FL. Estudos recentes analisam como as condições da rede afetam o desempenho do modelo [Kairouz et al. 2021]. Ao mesmo tempo, a configuração das aplicações de FL impacta a carga de trafego FL introduzida na rede. Assim, é necessário buscar um equilíbrio entre as variáveis que modificam a carga da rede a fim de otimizar o desempenho do modelo.

Em [Eriş et al. 2021] foi avaliado o impacto do trafego de fundo no desempenho do modelo. Foi simulado tráfego de FL utilizando o protocolo UDP. O tráfego de fundo foi simulado seguindo uma distribuição de Poisson, de forma a competir pela largura de banda disponível. Os resultados da simulação mostraram que o tráfego de fundo aumenta a perda de pacotes, a qual prejudica significativamente o desempenho do FL, especialmente em condições de congestionamento. Isso destaca o papel crítico das condições de rede na convergência e na acurácia do modelo FL.

Em [Rodio et al. 2023] foram abordados os desafios do suporte do treinamento federado em redes sem fio com perdas de pacotes. O trabalho propõe considerar perdas heterogêneas de pacotes entre os clientes. Dando maior peso aos clientes que demoraram mais rodadas em enviar com sucesso os seus parâmetros ao servidor. Foi proposto uma abordagem que mitiga os efeitos das perdas de pacotes sem aumentar o consumo de recursos. O trabalho conclui que uma alta confiabilidade de transmissão não é um fator necessário para o treinamento federado.

Em [Tedeschini et al. 2023] foi modelado o tráfego FL em redes sem fio com orquestração assíncrona e clientes heterogêneos. O modelo proposto ressalta a necessidade de adaptar os parâmetros da aplicação de FL para o ambiente no qual será colocada. O artigo também apresenta um algoritmo para selecionar o número ideal épocas de treinamento local feitas por cada cliente antes de compartilhar os pesos com o servidor. Tal abordagem, busca equilibrar a largura de banda com o tamanho da base de dados do cliente e o seu poder de processamento.

Em [Paolini et al. 2024] é proposto a integração de Fountain Codes (FC) para mitigar os efeitos da perda de pacotes nas atualizações dos modelos. FC é uma classe de códigos de correção de erros usado principalmente em sistemas de comunicação e armazenamento de dados. O uso de FC demonstraram melhorias significativas em condições de alta perda de pacotes, típicas de redes sem fio instáveis. Ao empregar FC e cumulative acknowledgment (ACK), o sistema pode recuperar dados perdidos e melhorar transmissão dos parâmetros do modelo. No entanto, as vantagens dos FC diminuem em ambientes

de baixa perda de pacotes, onde a sobrecarga computacional dos FC torna-se não negligenciável. Isso ressalta a necessidade de abordagens adaptativas, como a seleção do protocolo mais adequado com base nas condições da rede.

Alguns trabalhos propõem uma análise sobre as diferentes tecnologias de rede; dadas suas peculiaridades, como latência e largura de banda [Li et al. 2020, Ciceri et al. 2022]. Esses estudos propõem mecanismos de fatiamento de largura de banda para redes Passive Optical Network (PONs), com foco na reserva de largura de banda e multiplexação para atender às demandas de rede do FL. Essa abordagem destaca a importância de um provisionamento de Quality of Service (QoS) personalizado para melhorar a comunicação nos sistemas FL. Esses artigos também identificam os desafios enfrentados pelo FL devido à natureza compartilhada dos recursos de rede, como em PONs.

Ainda, dando suporte aos trabalhos anteriores no estudo do impacto das condições da rede no desempenho dos modelos federados, diversos simuladores têm sido utilizados com essa finalidade. Uma discussão comparativa dos simuladores utilizados na avaliação do desempenho da federação revela que, enquanto alguns simuladores, como o FedSim [Varno 2022] e o FedML [He et al. 2020], focam especificamente nos algoritmos de aprendizado federado, eles apresentam escassa habilidade para configurar a rede. Por outro lado, simuladores específicos de rede, como o NS-3 [Riley and Henderson 2010], OMNeT++ [Varga 2010] e Mininet [Kaur et al. 2014], oferecem uma maior flexibilidade na modelagem das condições da rede, mas necessitam de implementações específicas para o ambiente federado. Nesse sentido, nenhum dos simuladores oferece a possibilidade de alterar os parâmetros da rede para uma mesma configuração da federação, sendo identificado um espaço para o desenvolvimento de um simulador baseado em *traces*. Dessa forma, para um mesmo experimento de aprendizado federado será possível observar seu comportamento em diferentes condições da rede.

Os artigos vistos nesta revisão detectam a importância do estudo da carga gerada na rede pelos modelos FL. Porém, as abordagens propostas neles focam em tecnologias e protocolos de comunicação específicos. Deixando de lado uma análise mais geral da relação que existe entre os tempos de comunicação e computação dos modelos. Nesse sentido, este estudo propõe uma metodologia para a avaliação de desempenho da rede FL, considerando a relação entre o tempo de computação e o tempo de comunicação. Para isso, foi criado um simulador que, usando filas de background e *traces*, permite uma visão em alto nível do sistema, permitindo, assim, comparar como os modelos e seus parâmetros impactam no desempenho da rede de comunicação e do modelo de FL.

3. Modelo do Sistema

Para garantir um processo independente do *hardware* disponível, propomos um *pipeline* que integra a emulação do treinamento FL com um simulador de acesso à rede desenvolvido internamente, o qual permite avaliar o impacto da carga de trabalho do FL . O fluxo do nosso pipeline é apresentado na Figura 2. No estágio 1, são escolhidas as aplicações FL que serão emuladas, a arquitetura do modelo que será treinado e os hiper-parâmetros do treinamento, como número de clientes, épocas e rodadas de treinamento. No estágio 2, a aplicação FL é emulada localmente usando o benchmark LEAF. Utilizamos o LEAF porque ele gera a quantidade de Operações de Ponto Flutuante (FLOPs) a cada rodada de treinamento por cliente. No entanto, essa implementação pode ser realizada com qualquer framework que emule aplicações FL. No estágio 3, o número de FLOPs utilizados para o treinamento dos modelos locais, é convertido no tempo de chegadas dos pacotes FL na rede. Após isso, no estágio 4, estes tempos de chegada são utilizados como entrada para o simulador de *traces*. O simulador realizará a simulação do fluxo de pacotes dos clientes ao CS e adicionará um tráfego concorrente ao fluxo FL. Por fim, no estágio 5, as métricas de modelo e da rede como precisão e atraso médio no envio dos modelos locais ao CS são coletadas.

O *benchmark* LEAF é um emulador que permite avaliar o comportamento do treinamento de aplicações FL (*i.e.,* Estágios 1 e 2). A ferramenta permite variar a configura-

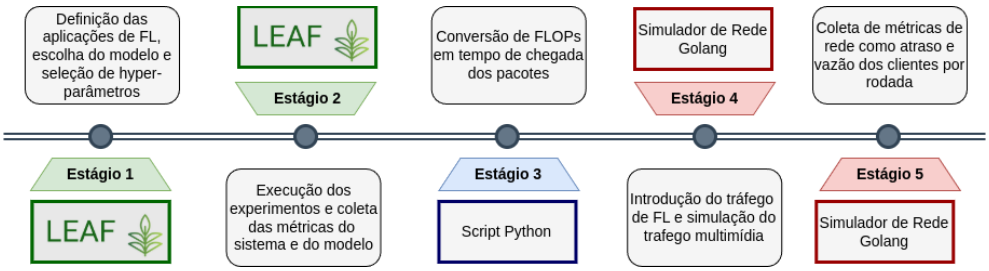


Figura 2. Pipeline para a Integração do Tráfego de Aplicações FL em um Simulador de Acesso à Rede baseado em Eventos Discretos.

ção da aplicação de FL que será emulada, que inclui o *dataset*, a arquitetura do modelo, o número de clientes, a taxa de aprendizado, o número de épocas e a fração dos dados locais usados para o treinamento (*minibatch*). O emulador fornece cinco aplicações FL, que abrangem tarefas de processamento de imagens (Federated EMNIST (FEMNIST), Celeba) e processamento de texto (Shakespeare, Reddit e Twitter), além de possibilitar a criação de aplicações personalizados. O dataset é particionado não Independente nem Identicamente Distribuída (nIID) entre os clientes. Para a agregação dos modelos, a ferramenta utiliza o algoritmo Federated Average (FedAvg). O processo de emulação é realizado localmente, portanto o treinamento dos modelos locais de cada cliente e a agregação no CS são realizados na mesma máquina.

Após o fim do treinamento FL, o LEAF escreve todas as métricas coletadas durante o treinamento em dois arquivos Valores Separados por Vírgula (CSV), um para as métricas do modelo e outro para as métricas sistema. No arquivo de métricas do modelo são armazenados os valores de acurácia, perda dos modelos, número de amostras para treinamento e partição de avaliação (treinamento e teste). Essas métricas são coletadas a cada número fixo de rodadas da treinamento. Adicionalmente, no arquivo de métricas do sistema, são armazenados para cada cliente em cada rodada de treinamento os valores de número de FLOPs, tamanho do modelo, quantidade de *bytes* enviados para o servidor e número de amostras empregadas para treinamento local.

Para a criação do arquivo de trilhas, foi implementado um *script* que converte os FLOPs em tempo de chegada de pacotes FL (*i.e.,* Estágio 3). O *script* recebe como entrada o arquivo de métricas do sistema, a quantidade de FLOPs dos cliente e tem como saída um arquivo de trilhas que contém o tempo de chegada e o tamanho de cada pacote enviado pelos clientes FL ao CS em cada rodada de treinamento. Durante o processo de conversão, foi considerado que a arquitetura dos modelos possui índices de paralelização diferentes, o que significa que cada modelo terá um tempo de processamento diferente para a mesma quantidade de processadores disponíveis. A eficiência da paralelização foi modelada com base na Lei de Amdahl [Amdahl 1967] que relaciona a porcentagem da tarefa que pode ser executada em paralelo com a quantidade de processadores disponíveis para execução, fornecendo o aumento de velocidade esperado com a paralelização do processo. Esta lei é definida por S(N) = $1/(1-P+\frac{P}{N})$, onde $S(N)$ é aumento de velocidade esperado (*speed-up*), N é o número de processadores disponível para a realização da tarefa e P é a porcentagem do código que pode ser processada em paralelo.

3.1. *TraceFL-Net-Sim*: um simulador de acesso à rede para aprendizado federado baseado em *Traces*

Desenvolvemos um simulador de código aberto para avaliar o impacto do tráfego FL e background na carga do acesso à rede, chamado de **Trace**-driven **F**ederated-Learning **Net**work **Sim**ulator (*TraceFL-Net-Sim*) [1]. Como pode ser visto na Figura 2, o simulador

[1] `https://github.com/wocn-unicamp/TraceFL-Net-Sim`

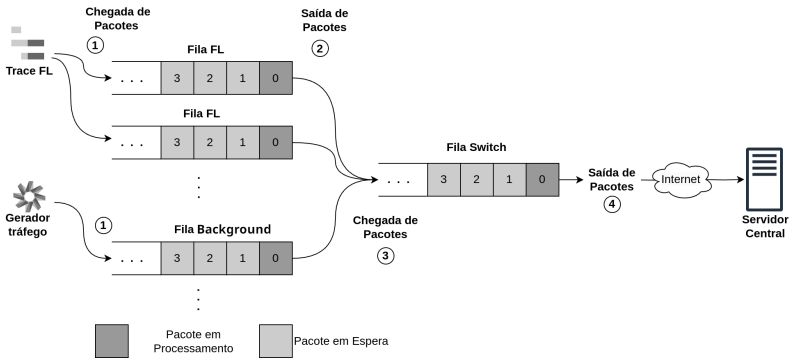


Figura 3. Arquitetura do simulador de eventos discretos *TraceFL-Net-Sim*.

utiliza os resultados gerados pelo LEAF e processados pelo *script* como arquivo de registro de eventos para gerar o tráfego FL. O simulador gera um arquivo que contém métricas de rede, tais como vazão média e atraso médio do tráfego FL. O *TraceFL-Net-Sim* utiliza uma simulação de eventos discretos para o fluxo de quadros *Ethernet* entre os nós da rede. Os pacotes de FL são fragmentados em N quadros no início da simulação e o pacote FL é reconstruído quando todos os N quadros alcançarem o CS. Ademais, a chegada dos quadros de tráfego de background ao buffer dos nós segue uma distribuição de Poisson, enquanto o tamanho de cada quadro segue uma distribuição uniforme, em consonância com as convenções da Teoria das Filas que viabilizam a derivação de modelos analíticos para o sistema.

TraceFL-Net-Sim utiliza quatro tipos de eventos para simular o fluxo de quadros na rede, como representado na figura 3. *(i)* chegada de quadros nas filas dos clientes FL e M/U/1 (*ARRIVAL_Q*); *(ii)* saída de quadros das filas dos clientes FL e M/U/1 (*DEPARTURE_Q*); *(iii)* chegada de quadros na fila da rede de acesso (*ARRIVAL_A*); e *(iv)* saída de quadros na fila da rede de acesso (*DEPARTURE_A*). Todos os quadros gerados pelo arquivo de registro e os quadros gerados de tráfego de background, fluem das filas dos respectivos nós (clientes FL e background) para a fila do switch na camada de enlace. Finalmente, o Switch envia os quadros para alcançar o CS. Foi assumido que o atraso entre o Switch e o CS é constante, para focar no impacto do tráfego FL na rede de acesso compartilhada pelos clientes. Por fim, foi considerado que todas as filas na simulação possuem *buffer* infinito, buscando facilitar a derivação de modelos analíticos e permitir simulações mais robustas.

Por outro lado, o processo de fragmentação gera um alto consumo de memória para a máquina responsável por executar a simulação. Modelos ML possuem uma alta variabilidade de tamanhos, saindo da ordem dos KBs até GBs, como no caso do treinamento de modelos de linguagem como BERT [Devlin et al. 2018] e Llama [Abhinav et al. 2024]. Ademais, este problema é agravado pela carga de trabalho imposta pelo tráfego de background. Por isso, o simulador foi implementado na linguagem de programação Go, aproveitando seus recursos de gerenciamento de concorrência e memória, já validados em aplicações como Docker [Docker 2025] e Ollama [Ollama 2025], os quais exigem um maior controle do consumo de memória e processamento.

Por fim, a métrica de atraso médio para a chegada dos modelos é calculada ao fim de cada rodada de treinamento. Para calcular o atraso do pacote FL é feita a diferença entre a chegada do primeiro quadro no buffer do dispositivo de rede até o último quadro do pacote FL que chega no CS. Ao fim da simulação, o *TraceFL-Net-Sim* gera dois arquivos CSV, de forma análoga ao LEAF. Um arquivo contendo as métricas de atraso médio por cliente durante as rodadas e um arquivo de métricas que armazena os valores da carga de trabalho FL e background e os atrasos dos clientes em cada fila do sistema simulado. O simulador foi validado com base na implementação do simulador de redes ópticas em [Ciceri et al. 2022], com foco na validação das implementações de geração de tráfego,

tanto de background como de aplicações FL.

4. Avaliação de Desempenho

Esta seção apresenta uma análise detalhada do desempenho de sistemas de FL sob diferentes parâmetros. Para a simulação do acesso à rede e do tráfego de background, utilizou-se o simulador de rede *TraceFL-Net-Sim*, desenvolvido internamente em Golang. Além disso, o benchmark LEAF [Caldas et al. 2018] foi empregado para a geração do tráfego de FL. O *TraceFL-Net-Sim* implementa a politica First-Come-First-Served (FCFS) para definir a ordem de enfileiramento dos quadros Ethernet na filas dos dispositivos de rede. Nossa ferramenta para simular uma rede de acesso com tráfego FL e background foi validada de forma extensiva.

4.1. Modelo de simulação e configuração

O cenário de simulação considera uma rede de acesso de banda larga, composta por um único dispositivo de acesso (*i.e.,* switch da camada de enlace) que atende um conjunto de $\mathcal{N}$ nós de rede distribuídos com topologia em árvore, conforme apresentado na Figura 1. Desses nós, um subgrupo $\mathcal{N}_\mathcal{F}$ suporta o tráfego FL, de aqui em diante nós FL, enquanto outro subgrupo $\mathcal{N}_\mathcal{B}$ suporta tráfego de *background*, de aqui em diante nós convencionais, tal que $\mathcal{N}_\mathcal{F} \subset \mathcal{N}$ e $\mathcal{N}_\mathcal{B} \subset \mathcal{N}$, com $\mathcal{N}_\mathcal{F} \cup \mathcal{N}_B = \mathcal{N}$ e $\mathcal{N}_\mathcal{F} \cap \mathcal{N}_B = \varnothing$. No cenário de simulação adotado, consideramos clientes homogêneos, nos quais cada cliente possui a mesma capacidade computacional de FLOPs, definida como $f_i = 250 \times 10^6$.

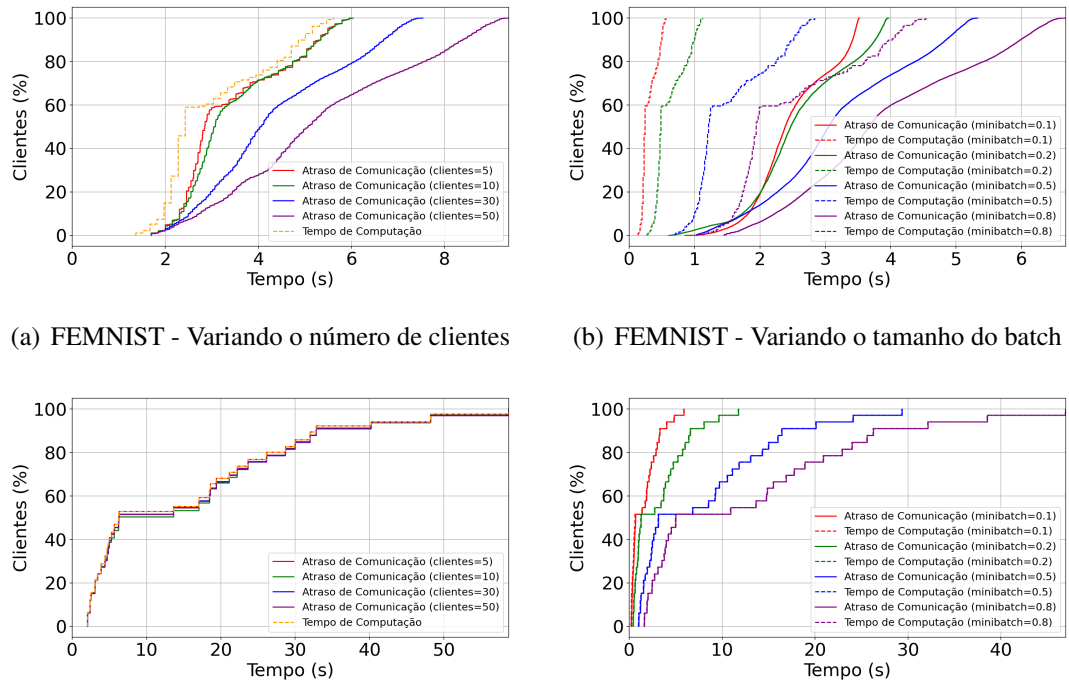
Foi utilizada uma largura de banda garantida de $b_i = 1.5$ Gbps para cada um dos nós de rede $i \in \mathcal{N}$, totalizando uma capacidade total do canal de *upstream* de $\sum_{i \in \mathcal{N}} b_i$. Além disso, o *switch* está conectado à internet por meio de um acesso de 2.25 Gbps.

A carga oferecida para os nós convencionais é $1.0 \cdot b_i$ enquanto a carga oferecida agregada para os nós FL depende da configuração dos parâmetros da aplicação. Foram variados tanto o número de clientes federados ($|\mathcal{N}_\mathcal{F}| = \{ 5, 10, 30, 50\}$) quanto o valor do minibatch (0.1, 0.2, 0.5 e 0.8). Além disso, foi simulada uma única carga que representa os nós convencionais ($|\mathcal{N}_\mathcal{B}| = 1$). O tráfego dos nós convencionais foi modelado a partir de uma distribuição de Poisson. Enquanto, o benchmark LEAF [Caldas et al. 2018], foi utilizado para gerar o tráfego de FL.

No caso do tráfego de FL, foram avaliadas duas aplicações. Uma aplicação para classificação de imagens e outra para previsão do próximo caracter, as quais utilizam os conjuntos de dados FEMNIST e Shakespeare, respectivamente. A primeira aplicação emprega uma Convolutional Neural Network (CNN) com duas camadas de convolução 5×5 para o treinamento do modelo, enquanto a segunda utiliza uma Long Short-Term Memory (LSTM) de duas camadas de 256 unidades cada. Os clientes de FL geraram 26.4 MB e 32.72 MB de dados a cada rodada de treinamento, empregando a CNN e o LSTM, respectivamente.

O algoritmo FedAvg foi empregado para agregar os parâmetros locais no servidor para as dois aplicação. Outras configurações do processo de aprendizado, como taxa de aprendizado e numero de épocas, seguiram as definições estabelecidas em [Li et al. 2021]. Para converter a quantidade de FLOPs em tempo de processamento, foi utilizada uma porcentagem da tarefa paralelizável de 0.95 e núcleos com capacidade de processamento de 0.25 GFLOPs/s.

Além disso, as nós de rede fragmentaram os modelos locais em quadros de acordo com o protocolo Ethernet, que possui uma Unidade Máxima de Transmissão de 1500 Bytes e um campo de cabeçalho para sinalização de 18 Bytes. Os comprimentos dos quadros para o tráfego convencional são distribuídos uniformemente entre 64 e 1518 Bytes. Cada cenário de simulação teve duração de 1000 rodadas de treinamento para o FEMNIST e 100 rodadas para o Shakespeare.



(a) FEMNIST - Variando o número de clientes

(b) FEMNIST - Variando o tamanho do batch

(c) Shakespeare - Variando o número de clientes

(d) Shakespeare - Variando o tamanho do batch

Figura 4. Percentual de clientes envolvidos na agregação versus tempo, utilizando o conjunto de dados FEMNIST e Shakespeare.

Em resumo, foram simuladas duas aplicações com diferentes configurações de parâmetros em uma rede de acesso, na qual os nós possuem uma largura de banda garantida. O tráfego de background foi modelado com uma carga alta, enquanto o tráfego de aprendizado federado foi ajustado conforme as configurações da aplicação. Dessa forma, é possível avaliar o desempenho da rede sob diferentes condições de tráfego, analisando o impacto das variáveis de configuração da aplicação no tempo de processamento e no atraso do tráfego FL.

4.2. Resultados da simulação

As Figuras 4 apresentam a proporção de clientes envolvidos na agregação em função do tempo para os datasets FEMNIST e Shakespeare, respectivamente. O tempo de computação representa o tempo mínimo de sincronização por rodada, desconsiderando qualquer atraso de comunicação. Além disso, as imagens à esquerda mostram o impacto da variação do número de clientes, enquanto os gráficos à direita ilustram o impacto do tamanho do batch na proporção de clientes envolvidos na agregação.

Observou-se que o tempo de computação permanece constante em relação ao número de clientes (Figuras 4(a) e 4(c)), enquanto o tempo de computação aumenta à medida que o tamanho do minibatch aumenta (Figuras 4(b) e 4(d)). Esse comportamento é consequência de que o número de amostras com as quais cada cliente treina o modelo é o mesmo, independentemente do número de clientes que participam do treinamento. Por outro lado, o tamanho de minibatch maior resulta em mais mostras sendo processados em cada cliente, exigindo mais tempo computacional para realizar o treinamento.

No caso do FEMNIST, um maior número de clientes leva ao congestionamento da rede, aumentando a latência (Figura 4(a)), enquanto no Shakespeare esse efeito é mitigado pela maior duração do treinamento (Figura 4(c)). Além disso, o tamanho do minibatch impacta diretamente o tempo de computação, mas seu efeito sobre o atraso de comunica-

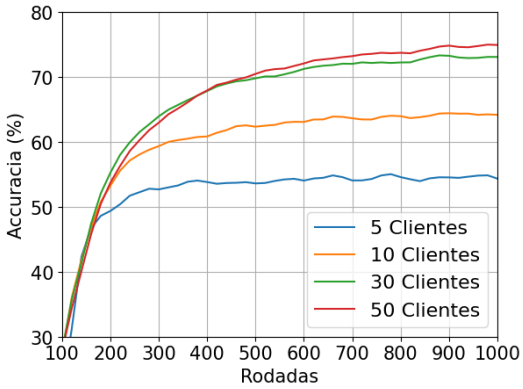


Figura 5. Acurácia de Treino do FEMNIST variando o número de clientes.

ção depende das características do dataset. No FEMNIST, minibatches menores resultam em maior atraso devido ao aumento na frequência de atualizações (Figura 4(c)), enquanto no Shakespeare a comunicação não se torna um fator limitante. Esses achados destacam a necessidade de um balanceamento cuidadoso entre a configuração das aplicações de FL e a quantidade de recursos disponíveis, uma vez que há um trade-off entre a carga computacional e a latência de comunicação gerada por essas configurações.

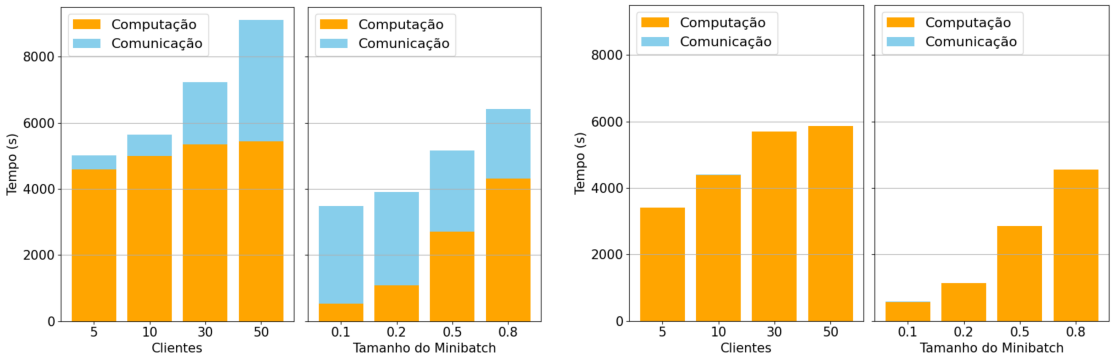
A Figura 5 mostra a acurácia do modelo federado durante as rodadas de treinamento no conjunto de dados FEMNIST, utilizando o algoritmo FedAvg. Ao variar o número de clientes selecionados para participar da federação em diferentes experimentos, observa-se uma tendência de melhora na acurácia do modelo à medida que o número de clientes aumenta. Isso ocorre porque a maior quantidade de clientes contribui para uma atualização mais diversificada dos gradientes, o que permite ao modelo generalizar melhor sobre os dados distribuídos.

As Figuras 6(a) e 6(b) apresentam o tempo total de treinamento, respectivamente, para os conjuntos de dados FEMNIST e Shakespeare. Para cada um dos modelos treinados com esses conjuntos de dados, varia-se o número de clientes e o tamanho do minibatch. O tempo total de treinamento é calculado somando o tempo de todas as rodadas de treinamento, distinguindo entre o tempo de computação e o tempo de comunicação do último cliente de cada rodada.

Observa-se que um maior número de clientes tende a aumentar o atraso de comunicação, ao mesmo tempo que se mantém o tempo de computação do modelo. Por outro lado, o aumento do tamanho do minibatch leva a um aumento significativo no tempo de computação, enquanto o tempo de comunicação diminui. Esta relação entre o tempo de computação e o tempo de comunicação se dá pelo fato de que, diante de uma distribuição mais esparsa dos tempos de computação, a carga sobre a rede na federação é reduzida, permitindo que o envio do modelo seja mais rápido. Assim, no caso do conjunto de dados Shakespeare, os últimos clientes a concluir a rodada encontram a rede ociosa, dispondo de toda a largura de banda disponível para enviar o modelo de forma mais rápida.

5. Conclusão

O treinamento de modelos de FL demanda um alto consumo de recursos de rede devido ao grande tamanho dos modelos e à alta frequência das atualizações, o que pode resultar em congestionamento em ambientes com largura de banda limitada e capacidade de processamento reduzida. A configuração dos parâmetros das aplicações, como hiperparâmetros, tipo de rede neural e até mesmo a própria aplicação, impacta diretamente a carga de tráfego gerada. Este artigo propôs uma metodologia para avaliar o impacto de configurações de FL no acesso à uma rede de acesso de banda larga. Essa metodologia integra o *TraceFL-Net-Sim*, um simulador de acesso à rede desenvolvido em Goland para



(a) FEMNIST (b) Shakespeare

Figura 6. Tempo total de treinamento.

este estudo, e o *benchmark* LEAF, onde o treinamento dos modelos é realizado no LEAF. Nos experimentos, observou-se um aumento no atraso de rede dos pacotes à medida que o número de clientes e o tamanho dos *batches* de treinamento aumentam. Ademais, o incremento dessas duas variáveis resulta em uma melhora na precisão do modelo treinado, indicando um compromisso entre o desempenho do modelo e o tempo de treinamento. Como trabalhos futuros, pode-se realizar uma avaliação do desempenho dos modelos em um cenário de FL síncrono, considerando o impacto do tempo de sincronização na quantidade de clientes envolvidos na agregação.

Agradecimentos

Pesquisa parcialmente financiada pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (proc. 2024/07007-5 e 23/00673-7), pelo Programa de Incentivo a Novos Docentes (proc. 3406/23) e Auxílio Início de Carreira (proc. 3098/24) da Unicamp, pela Fundação de Desenvolvimento da Unicamp (FUNCAMP) (proc. 43783-24 e 23886-25), pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) proc. 405940/2022-0, pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) — Código de Financiamento 88887.954253/2024-00, 88887.977361/2024-00, 88887.005653/2024-00, 88887.955381/2024-00.

Referências

[Abhinav et al. 2024] Abhinav, D., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[Amdahl 1967] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), page 483–485, New York, NY, USA. Association for Computing Machinery.

[Caldas et al. 2018] Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2018). Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

[Ciceri et al. 2022] Ciceri, O. J., Astudillo, C. A., Zhu, Z., and da Fonseca, N. L. (2022). Federated learning over next-generation ethernet passive optical networks. *IEEE network*, 37(1):70–76.

[Cohen et al. 2017] Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE.

[Devlin et al. 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Docker 2025] Docker, I. (2025). Docker. `https://www.docker.com/`. Acessado em 27 mar. 2025.

[Eriş et al. 2021] Eriş, M. C., Kantarci, B., and Oktug, S. (2021). Unveiling the wireless network limitations in federated learning. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 262–267. IEEE.

[He et al. 2020] He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., et al. (2020). Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*.

[Jin et al. 2023] Jin, W., Yao, Y., Han, S., Gu, J., Joe-Wong, C., Ravi, S., Avestimehr, S., and He, C. (2023). Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system. *arXiv preprint arXiv:2303.10837*.

[Kairouz et al. 2021] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210.

[Kaur et al. 2014] Kaur, K., Singh, J., and Ghumman, N. S. (2014). Mininet as software defined networking testing platform. In *International conference on communication, computing & systems (ICCCS)*, pages 139–42.

[Li et al. 2021] Li, J., Chen, L., and Chen, J. (2021). Scalable federated learning over passive optical networks. In *2021 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3.

[Li et al. 2020] Li, J., Shen, X., Chen, L., and Chen, J. (2020). Bandwidth slicing to boost federated learning over passive optical networks. *IEEE Communications Letters*, 24(7):1492–1495.

[McMahan et al. 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

[Ollama 2025] Ollama, I. (2025). Ollama. `https://ollama.com/`. Acessado em 27 mar. 2025.

[Paolini et al. 2024] Paolini, E., Pinto, A., Valcarenghi, L., Andriolli, N., Maggiani, L., and Esposito, F. (2024). Efficient distributed learning over lossy wireless networks. In *2024 20th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE.

[Parliament 2016] Parliament, E. (2016). Regulation (eu) 2016/679 of the european parliament and of the council. *Official Journal of the European Union*.

[Riley and Henderson 2010] Riley, G. F. and Henderson, T. R. (2010). The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer.

[Rodio et al. 2023] Rodio, A., Neglia, G., Busacca, F., Mangione, S., Palazzo, S., Restuccia, F., and Tinnirello, I. (2023). Federated learning with packet losses. In *2023 26th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 1–6. IEEE.

[Shakespeare 2014] Shakespeare, W. (2014). *The complete works of William Shakespeare*. Race Point Publishing.

[Tedeschini et al. 2023] Tedeschini, B. C., Savazzi, S., and Nicoli, M. (2023). A traffic model based approach to parameter server design in federated learning processes. *IEEE Communications Letters*, 27(7):1774–1778.

[Varga 2010] Varga, A. (2010). Omnet++. In *Modeling and tools for network simulation*, pages 35–59. Springer.

[Varno 2022] Varno, F. (2022). Fedsim: A generic federated learning simulator.