Uma Perspectiva Analítica para Avaliação de Desempenho no Aprendizado Federado

Francisco Airton Silva[1], Allan M. de Souza[3], Iago Almeida[1], Eduardo Cerqueira[2], Luiz Fernando Bittencourt[3] e Denis Rosário[2]

[1]Universidade Federal do Piauí – PI – Brasil

[2]Universidade Federal do Pará – PA – Brasil

[3]Universidade de Campinas – SP – Brasil

{faps,iago.almeida}@ufpi.edu.br, {cerqueira,denis}@ufpa.br

{bit,allanms}@ic.unicamp.br

Resumo. *Implementar e avaliar o desempenho de soluções de Aprendizado Federado (FL) em ambientes reais ou simulados apresenta desafios significativos, não apenas devido a limitações de escalabilidade, tempo e custo, mas também à heterogeneidade estatística e sistêmica, incluindo restrições de rede e problemas de desempenho, que são difíceis de reproduzir em cenários experimentais ou simulados. Nesse contexto, os modelos analíticos emergem como uma abordagem poderosa para compreender o comportamento de sistemas de FL em larga escala e densos. Esses modelos permitem uma análise mais detalhada e eficiente, ao integrar cenários densos com diferentes opções de conectividade, por exemplo, intermitente, e incorporar a variabilidade estocástica em transições e estados, capturando de forma realista a complexa dinâmica de sistemas distribuídos. Este artigo apresenta a modelagem de Stochastic Petri Net (SPN) como uma abordagem analítica complementar aos simuladores de FL tradicionais em cenários dinâmicos e densos. O modelo SPN permite a simulação detalhada da dinâmica FL principal, incluindo participação assíncrona, seleção de clientes e utilização de recursos. A aplicação do modelo de SPN para análise de FL é ilustrada, calibrada e validada com um estudo de caso utilizando dados de um experimento real. Os resultados fornecem insights importantes para a otimização de sistemas distribuídos em diferentes domínios.*

Abstract. *Conducting Federated Learning (FL) in real-world environments poses significant challenges due to client heterogeneity, network constraints, and scalability issues, which are difficult to reproduce in experimental settings. In this context, analytical models offer a more comprehensive and efficient view of the behavior of large-scale FL systems, and thus it is important to model dense scenarios and intermittent connectivity efficiently, integrating stochastic variability in transitions and states. In this work, we introduce a Stochastic Petri Net (SPN) modeling as a complementary analytical approach to traditional FL simulators. The proposed SPN model enables detailed simulation of key FL dynamics, including asynchronous participation, device dropouts, and resource contention. The application of the SPN model is showcased using experiments from a testbed. The results provide important insights for the optimization of distributed systems in different domains.*

1. Introdução

O Aprendizado Federado (do inglês, *Federated Learning* - FL) permite o aprendizado de máquina colaborativo em dispositivos distribuídos [Banabilah et al. 2022]. Ao treinar modelos localmente em dispositivos da Internet das Coisas (IoT) e agregar as atualizações do modelo treinado em um servidor, o FL oferece uma abordagem descentralizada para o aprendizado de máquina que minimiza a transmissão de dados ao mesmo tempo que mantêm a privacidade dos dados. Apesar de seu potencial, o FL apresenta desafios únicos em comparação com o aprendizado de máquina centralizado [Wen et al. 2023]. Por exemplo, a variabilidade dos dispositivos participantes que tem diferentes capacidades de processamento e conectividade, impactando o tempo de resposta e sincronização durante o treinamento. Outro desafio relevante envolve a eficiência de agregação, já que a participação irregular dos clientes e a qualidade variável das atualizações exigem estratégias adaptativas.

Realizar experimentos com FL em ambientes reais é um processo complexo, demorado e de alto custo [Bonawitz 2019]. A necessidade de configurar diversas variáveis do sistema aumenta a complexidade, dificultando a realização de testes em diferentes cenários e condições de rede ou de clientes. Por exemplo, em ambientes que requerem participação assíncrona dos clientes, onde dispositivos entram e saem de forma imprevisível, a execução de atualizações sincronizadas se torna especialmente desafiadora. Além disso, restrições de rede, como latências variáveis, limitações de largura de banda e padrões de comunicação inconsistentes, impõem obstáculos significativos, impactando diretamente a eficiência e o desempenho do sistema de FL. Além disso, a natureza dinâmica das populações de clientes representa um desafio adicional significativo, pois o número de dispositivos ativos pode variar de forma imprevisível, exigindo uma escalabilidade adaptável que é complexa de implementar e gerenciar. Outro ponto crítico são os problemas de simultaneidade e os potenciais gargalos, que se tornam mais evidentes à medida que as cargas dos clientes aumentam, frequentemente resultando em sobrecarga do servidor e limitação de recursos, dificuldades que não são facilmente antecipadas em configurações simuladas. Para enfrentar esses desafios, experimentos de FL no mundo real exigem, em primeiro lugar, métodos de simulação capazes de capturar de maneira eficaz cenários, requisitos, contextos e comportamentos reais.

Os simuladores de FL são projetados principalmente para facilitar a experimentação de algoritmos de FL em ambientes controlados, embora limitado [Li et al. 2021]. Por exemplo, simuladores populares como FLSim [Kim et al. 2021], FedJAX [Gilmer et al. 2021], FLUTE [Joshi et al. 2021], Flower [Beutel et al. 2020] e TFF [TensorFlow Federated 2021] foram criados para dar suporte a FL, oferecendo diversos recursos [Li et al. 2020]. Esses simuladores permitem que pesquisadores estudem a dinâmica do FL, otimizem protocolos e testem novos algoritmos em condições simuladas que se aproximam de ambientes do mundo real [Kholod et al. 2020]. Os simuladores possuem pontos fortes, porém, também limitações. Os simuladores de FL apresentam limitações, incluindo a necessidade de recursos computacionais elevados para escalar experimentos [Kairouz et al. 2021], ajustes manuais e experimentação repetitiva para o teste de parâmetros [Savazzi et al. 2021]. Além disso, enfrentam limitações em termos de análise detalhada de gargalos sistêmicos, como congestionamento de rede e contenção de recursos, devido ao foco primário nas interações cliente-servidor.

Modelos analíticos, como cadeias de Markov, redes de Petri e modelos baseados

em filas, podem oferecer uma visão mais abrangente e eficiente do comportamento de sistemas de FL em larga escala. Esses modelos se baseiam em abstrações matemáticas que capturam dinâmicas complexas, como variações de latência, taxas de desconexão e restrições de recursos, sem a necessidade de simular diretamente cada entidade do sistema. Essa abordagem pode reduzir significativamente os custos computacionais e o tempo, permitindo a análise de cenários dinâmicos ou com recursos limitados, como aqueles encontrados em implementações de FL com dezenas de clientes ou conectividade intermitente. Dentre tais modelos, as Redes de Petri Estocásticas (SPNs) são modelos poderosos para representar sistemas de FL em larga escala, capturando dinâmicas complexas como taxas de desconexão e latências variáveis. As SPNs permitem modelar facilmente cenários com diversos de dispositivos IoT e diferentes formas de conectividade, inclusive intermitente o que é esperado em ambientes de cidades inteligentes, de forma eficiente, integrando variabilidade estocástica em transições e estados. Essa flexibilidade torna as SPNs ideais para analisar gargalos, interdependências e otimizar sistemas complexos e dinâmicos. Desta forma, percebe-se que é essencial entender o impacto de sistemas de FL sob condições realistas e complexa por meio de um modelo analítico.

Este artigo propõe um modelo baseado em SPNs para descrever e analisar sistemas de FL sob condições realistas. O modelo é capaz de capturar as principais etapas do protocolo de FL, nomeadamente, seleção, treinamento e agregação, integrando variabilidade estocástica para representar de forma precisa eventos como atrasos, desconexões e falhas de dispositivos em cenários dinâmicos e densos. A validade do modelo foi comprovada através de experimentos realizados em um testbed composto por dispositivos Raspberry Pi, que demonstraram sua precisão na reprodução de métricas essenciais, como o tempo médio por rodada e a taxa de conclusão de rodadas. Além disso, o modelo foi utilizado em estudos de caso para avaliar o impacto de parâmetros como tamanho de coorte, tempo limite de agregação e probabilidade de falha no desempenho do FL. Os resultados fornecem *insights* importantes para a otimização de sistemas distribuídos em diferentes domínios, como IoT e saúde, destacando o potencial das SPNs como ferramenta analítica e eficiente para modelar e simular problemas de FL com dezenas de dispositivos.

O restante deste artigo é organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta uma abordagem analítica baseada em SPN para avaliar sistemas FL. Especificamente, nesta seção é detalhado o protocolo de FL tradicional, o modelo de SPN proposto, incluindo as condições e métricas usadas para avaliar o desempenho do FL, bem como os resultados obtidos. Finalmente, a Seção 4 apresenta as conclusões deste trabalho e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Foram levantados seis artigos relacionados a este trabalho, abordando diferentes estratégias e modelos para otimizar e adaptar o FL em diversos contextos. Liu et al. [Liu et al. 2024] exploram o uso de Processos de Decisão de Markov (MDP) para otimizar a eficiência na agregação do FL em redes de borda, destacando métricas como precisão e tempo de treinamento. Já Albelaihi et al. [Albelaihi et al. 2021] propõem um modelo baseado em filas para selecionar de forma adaptativa os participantes em ambientes heterogêneos, priorizando o número suportado de clientes e o tempo médio de rodada como métricas de avaliação.

Outros trabalhos exploram a integração de blockchain no FL. Turgay et al. [Turgay 2022] utilizam um modelo de filas para adaptar o FL em sistemas de gestão

de saúde, destacando a integração com blockchain para melhorar a gestão de dados e estratégias baseadas em timeout. Wilhelmi et al. [Wilhelmi et al. 2021], por sua vez, propõem uma arquitetura sem servidor habilitada por blockchain para operações assíncronas no FL, utilizando cadeias de Markov para avaliar métricas como tempo médio de rodada e tempo de fila, com estratégias baseadas na precisão necessária.

Os dois últimos trabalhos focam em abordagens descentralizadas e dinâmicas. Du et al. [Du et al. 2022] propõem o uso de cadeias de Markov para estabelecer consenso descentralizado em redes IoT industriais, avaliando perda de validação e taxa de rodadas. Por outro lado, Pu et al. [Pu et al. 2024] abordam a adaptação dinâmica do FL para lidar com distribuições enviesadas de dados locais, utilizando cadeias de Markov para melhorar a precisão do modelo e adaptação da distribuição de dados. Esses trabalhos refletem a diversidade de estratégias e modelos empregados no FL, destacando a necessidade de avanços em agregação, escalabilidade e adaptação a condições heterogêneas.

Tabela 1. Comparação de abordagens de aprendizagem federada

Artigo	**Objetivo**	**Tipo de modelo**	**Métricas**	**Estratégia para iniciar agregação**
[Liu et al. 2024]	Otimizar a eficiência de agregação na aprendizagem federada via MDP	Cadeia de Markov	Precisão, tempo de treinamento	Baseada na precisão necessária
[Albelaihi et al. 2021]	Selecionar participantes ideais em ambientes de aprendizagem federada heterogêneos	Enfileiramento	Número de clientes suportados, tempo médio de rodada	Baseada no número de clientes
[Turgay 2022]	Adaptar o FL na área da saúde com integração de blockchain	Modelo de enfileiramento	Número de clientes suportados	Baseada no tempo limite
[Wilhelmi et al. 2021]	Habilitar o FL sem servidor usando blockchain para operação assíncrona	Cadeia de Markov	Tempo médio de rodada, tempo de fila	Baseada na precisão necessária
[Du et al. 2022]	Estabelecer consenso descentralizado usando a cadeia de Markov na IoT industrial	Cadeia de Markov	Perda de validação, taxa de rodada	Baseada no número de clientes
[Pu et al. 2024]	Adaptar oFL para distribuição de dados de cauda longa para melhorar a precisão do modelo	Cadeia de Markov	Adaptação da distribuição de dados, precisão do modelo	Baseada no número de clientes
Nossa proposta	Prever o desempenho do FL com foco em aspectos de comunicação e mecanismos de acionamento de agregação	Rede de Petri Estocástica	Taxa de rodada, tempo médio de rodada	Baseada no número de clientes e no tempo limite

A Tabela 1 sumariza algumas abordagens de FL, evidenciando os avanços e limitações de trabalhos recentes em termos de objetivos, modelos utilizados, métricas avaliadas e estratégias de agregação. O diferencial deste trabalho está na capacidade de superar as limitações das Cadeias de Markov, que apresentam dificuldade em capturar dependências complexas e interações simultâneas entre participantes em sistemas dinâmicos quando há muitos estados. Em particular, os modelos baseados em Cadeias de Markov assumem transições entre estados que dependem apenas do estado atual, ignorando influências históricas e paralelas que são fundamentais em ambientes heterogêneos e escaláveis. Isso restringe sua aplicabilidade em cenários com múltiplas variáveis interdependentes e alta variabilidade de comunicação. Diferentemente de abordagens que avaliam métricas restritas, como número de clientes suportados ou tempo de fila, o modelo

apresentado neste artigo incorpora taxa de rodada e tempo médio de rodada, fornecendo uma visão mais abrangente do desempenho do sistema. Além disso, a combinação de gatilhos baseados em contagem de clientes e limites de tempo permite maior adaptabilidade às variabilidades de disponibilidade e comunicação. Com isso, a abordagem apresentada neste artigo não apenas aborda as lacunas existentes, mas estabelece um novo padrão para a modelagem e otimização do FL em cenários complexos.

3. Uma Abordagem Analítica Com SPN para Avaliar Sistemas FL

Inicialmente, a Seção 3.1 descreve a arquitetura tradicional de FL, estabelecendo o contexto e os principais componentes do protocolo. Em seguida, a Seção 3.2 detalha o modelo SPN proposto, representando cada etapa do protocolo de FL por meio de lugares, transições e condições de guarda. A Seção 3.3 foca na validação do modelo SPN, comparando suas previsões com dados experimentais reais. Por fim, a Seção 3.4 extrapolando os resultados, utiliza-se o modelo SPN validado em um estudo de caso, explorando como diferentes parâmetros, como tamanho do coorte, timeout e probabilidade de falha, influenciam o desempenho do FL.

3.1. Arquitetura de FL

Para este trabalho foi considerada uma arquitetura de FL genérica tal como descrito na Figura 1, onde o FL é um processo iterativo dividido em interações cliente-servidor conhecidas como rodadas de aprendizagem [Bonawitz 2019]. Em cada rodada de comunicação, um subconjunto de dispositivos é selecionado pelo servidor central para receber o modelo global, também conhecido como fase de seleção de clientes do FL. Na fase de treinamento, cada cliente desse subconjunto obtém o modelo do servidor central e executa o treinamento com base em seus dados locais. Por fim, durante a fase de agregação, os clientes concluem seu treinamento local e apenas a atualização de um modelo local é enviada de volta a um servidor. Depois, uma determinada estratégia agrega os modelos locais compartilhados em servidores, resultando em um modelo global.

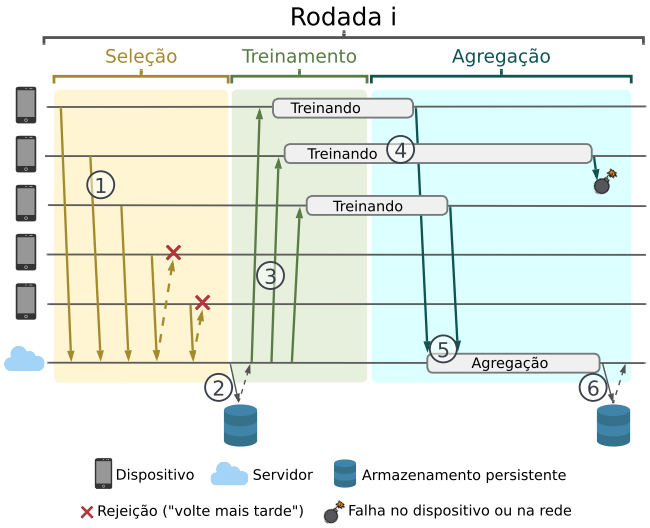


Figura 1. Arquitetura do FL tradicional [Bonawitz 2019]

Em uma abordagem de agregação semi-síncrona, somente os clientes que relatam os modelos dentro do tempo alocado são incluídos na agregação, ou seja, aqueles que não

respondem a tempo são desconsiderados sem afetar o protocolo geral. Se atualizações suficientes forem recebidas dentro do período designado, a rodada é concluída com sucesso e o modelo global do servidor é atualizado. Caso contrário, se o limite não for atingido, a rodada é abandonada. O protocolo inclui parâmetros como tempos limite e níveis mínimos de participação para gerenciar o tempo de cada fase de forma flexível, equilibrando a robustez contra as desistências do dispositivo com eficiência nas conclusões da rodada.

3.2. Modelo de SPN para uma Arquitetura de FL

Neste artigo é proposto um modelo SPN para avaliar sistemas de FL sob condições realistas e complexas, tal como pode ser observado na Figura 2. A Tabela 2 e 3 apresentam a descrição dos elementos do modelo. O SPN é um modelo matemático utilizado para descrever e analisar sistemas dinâmicos, composta por lugares, transições e arcos que conectam esses elementos. As transições disparam quando as condições especificadas são atendidas, tal como os dispositivos que estão disponíveis e atendem a determinados critérios. Após uma transição disparar, ela consome tokens dos lugares de entrada e os move para os lugares de saída, alterando o estado do sistema modelado [Maciel 2023]. Desta forma, o modelo de SPN proposto opera por meio de uma sequência estruturada de lugares e transições, que representam os estágios de participação do dispositivo nas fases de seleção, treinamento e agregação dentro de cada rodada do FL. Cada uma dessas fases está sincronizada para garantir a melhoria iterativa de um modelo global em um servidor FL centralizado.

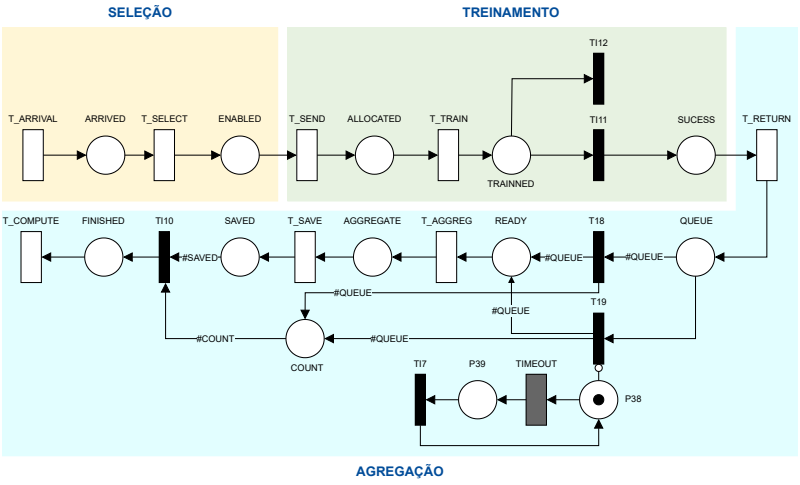


Figura 2. Modelo SPN para avaliar sistemas de FL.

Na fase de *Seleção* do FL, os dispositivos que estão disponíveis e atendem a determinados critérios, como status de carregamento e conectividade de rede, chegam ao local ARRIVED, onde a transição T_ARRIVAL significa o tempo entre as chegadas do cliente. Por outro lado, a transição T_SELECT, operando como um servidor infinito (operação paralela), permite que o servidor de agregação do FL selecione dispositivos qualificados, também chamados de clientes, e os mova para o local ENABLED, sinalizando prontidão para fase de treinamento.

Na fase de *Treinamento*, os clientes selecionados são configurados com os parâmetros mais recentes do plano e modelo FL. Por exemplo, a transição T_SEND,

Tabela 2. Lugares no modelo SPN

Componente(s	Descrição
ARRIVED	Dispositivos que sinalizaram sua disponibilidade para participar de uma rodada de aprendizado federado.
ENABLED	Dispositivos que foram selecionados e estão prontos para prosseguir para a fase de configuração.
ALLOCATED	Dispositivos alocados para realizar cálculos de treinamento local.
TRAINED	Dispositivos que concluíram o treinamento local em seus conjuntos de dados.
SUCCESS	Dispositivos que concluíram o treinamento local e relataram suas atualizações ao servidor.
QUEUE	Fila de dispositivos aguardando agregação ou retornando ao pool de seleção.
READY	Dispositivos preparados para agregação, tendo relatado com sucesso suas atualizações.
SAVED	Parâmetros do modelo agregados e salvos após combinar atualizações dos dispositivos que reportaram.
FINISHED	Dispositivos que concluíram todas as tarefas da rodada e foram marcados como finalizados.
COUNT	Conta o número de dispositivos que concluíram o relatório e estão prontos para agregação.

Tabela 3. Transições no modelo SPN

Componente(s)	Descrição	Semântica do Servidor
T_SELECT	Tempo gasto pelo servidor para selecionar dispositivos elegíveis para participação na rodada atual.	Servidor Infinito
T_SEND	Tempo para enviar o plano de FL e os parâmetros do modelo aos dispositivos selecionados.	Servidor Infinito
T_TRAIN	Duração alocada para os dispositivos realizarem o treinamento local.	Servidor Infinito
T_RETURN	Tempo permitido para que os dispositivos enviem suas atualizações de volta ao servidor.	Servidor Infinito
T_AGGREG	Tempo para agregar atualizações locais dos dispositivos e gerar um modelo global atualizado.	Servidor Único
T_SAVE	Tempo necessário para salvar o modelo global atualizado após a agregação.	Servidor Único
T_COMPUTE	Tempo gasto para cálculos finais e preparações para a próxima rodada.	Servidor Único
T_ARRIVAL	Chegada imediata de dispositivos prontos e disponíveis para aprendizado federado.	Servidor Único
T11 (SUCCESS)	Indica a conclusão bem-sucedida do treinamento local e do relatório por um dispositivo.	Servidor Único
T12 (FAIL)	Acionado quando um dispositivo falha em concluir o treinamento ou o relatório, potencialmente devido a desconexões.	Servidor Único
T18	Redistribui dispositivos na fila para agregação, preparando-os para o processamento final.	Servidor Único
T19	Redistribui dispositivos que precisam tentar novamente ou estão em um timeout devido à ausência de resposta.	Servidor Único
T17	Gerencia dispositivos que não concluíram o relatório a tempo, lidando com timeouts no protocolo.	Servidor Único
T10 (FINISH)	Marca a conclusão da rodada de aprendizado federado para todos os dispositivos relatando.	Servidor Único

também definida como um servidor infinito, distribui esses parâmetros do servidor para cada cliente. Uma vez configurados, os clientes são movidos para o local ALOCATED e ficam prontos para treinamento local. Os dispositivos no local ALOCATED realizam treinamento local em seus respectivos conjuntos de dados locais, e a transição T_TRAIN, funcionando como um servidor infinito, simula o tempo que cada dispositivo leva para concluir suas computações locais. Ao atingir o local TRAINED, há duas possibilidades, a saber: i) o processo de treinamento pode falhar (disparando a transição imediata TI12); ou ii) o processo de treinamento pode ter sucesso (disparando a transição imediata TI11). Essas duas transições somam 100% de probabilidade, e o avaliador deve configurar a probabilidade para os dois caminhos.

Por fim, na fase de *Agregação*, os dispositivos que concluíram seu treinamento relatam suas atualizações do modelo local de volta ao servidor FL. Isso é representado pela transição T_RETURN, que, como um servidor infinito, lida com o retorno de atualizações de vários dispositivos simultaneamente. No servidor, há uma fila representada pelo local QUEUE, onde o servidor deve lidar com os retornos recebidos. As condições de guarda especificadas no modelo SPN de FL garantem transições controladas com base no estado de determinados locais. Duas condições são implementadas

nas transições TI18 e TI19. A transição TI18 aplica o limite com base no limite de número do cliente para iniciar a agregação. Por outro lado, a transição TI19 aplica o limite com base em um tempo limite pré-especificado. Para a transição TI8, a condição $(\#QUEUE > 0)$ AND $(\#QUEUE =$ cohort) requer que o número de *tokens* no lugar 'QUEUE' seja maior que zero e corresponda ao tamanho do coorte para que ele seja acionado. Isso implica que o sistema espera até que um tamanho de grupo específico seja atingido em 'QUEUE' antes de prosseguir, garantindo o processamento em lote de dispositivos. TI9 aplica outra estratégia simultânea com base no tempo limite. A condição $(\#QUEUE > 0)$ permite que a transição seja acionada enquanto houver pelo menos um *token* em 'QUEUE' e o arco inibidor para colocar P38 significa que ele será acionado se a transição TIMEOUT for acionada. O TIMEOUT de transição é determinístico, o que significa que ele varia em um intervalo de tempo fixo. Este estágio termina com todos os clientes habilitados juntos no local READY.

O servidor de agregação do FL agrega as atualizações recebidas dentro de um período de tempo especificado. A transição T_AGGREG, definida como um único servidor (um processo sequenciado), executa esta tarefa de agregação, movendo o modelo agregado de cada cliente para o local AGREGATE uma vez concluído. Este modelo atualizado inclui contribuições apenas de dispositivos que relataram no passado, garantindo uma atualização robusta para o modelo global. Após a fase de agregação, o modelo é salvo e a rodada é marcada como concluída. As transições T_SAVE e T_COMPUTE, ambas servidores únicos, representam as etapas finais. Nelas, o modelo agregado é salvo e quaisquer cálculos necessários são realizados antes da próxima rodada. Dispositivos de locais SAVED são movidos para o local FINISHED juntos (#SAVED tokens transformados em apenas um), indicando a conclusão bem-sucedida da rodada. TI10 tem a condição $(\#SAVED > 0)$ AND $(\#SAVED = \#COUNT)$, que requer que o local SAVED tenha tokens e que o número de tokens em SAVED seja igual ao local 'COUNT'. Esta condição garante que todas as atualizações necessárias sejam salvas antes de finalizar o processo de agregação, mantendo a consistência na atualização do modelo global. Essas condições de proteção gerenciam o fluxo dentro do modelo, reforçando a sincronização e o processamento em lote quando necessário.

3.2.1. Métricas

Este artigo usa duas métricas principais para avaliar o sistema em questão: R_{rate} (taxa de rodadas) e T_{mean} (tempo médio da rodada), embora métricas adicionais também possam ser obtidas e incluídas facilmente no sistema. Essas métricas são derivadas usando princípios da teoria de filas. Especificamente, a Lei de Little [DeGlopper 1992], que afirma que o número médio de itens em um sistema de filas (L) é o produto da taxa de chegada (λ) e o tempo médio que um item passa no sistema (W):

$$L = \lambda W \tag{1}$$

No contexto do modelo de FL, R_{rate} representa a taxa efetiva de rodadas concluídas, a qual é calculada como:

$$R_{rate} = \frac{\mathbb{E}\{\#\text{FINISHED}\}}{\text{finish\_time}} \tag{2}$$

onde $\mathbb{E}\{\#\text{FINISHED}\}$ é o número esperado de rodadas concluídas, e *finish_time* é o tempo associado à transição T_COMPUTE, marcando o fim de uma rodada. Essa métrica fornece *insights* sobre o rendimento do modelo, indicando quantas rodadas são concluídas por hora.

Por outro lado, T_{mean} mede a duração média de uma única rodada, a qual reflete a latência por rodada, fornecendo *insights* sobre o tempo necessário para concluir cada ciclo no processo de FL. Desta forma, O T_{mean} é calculada como:

$$\begin{aligned} T_{mean} &= \frac{1}{R_{rate}} \\ &= \frac{\text{finish\_time}}{\mathbb{E}\{\#\text{FINISHED}\}} \end{aligned} \tag{3}$$

Juntas, essas métricas ajudam a avaliar a eficiência e a velocidade do sistema, destacando o equilíbrio entre a taxa de transferência e a duração individual da rodada na estrutura de FL. O modelo SPN fornece uma estrutura para simular e analisar o processo de FL, incluindo o impacto de interrupções e atrasos de dispositivos no desempenho geral. Ao incorporar semântica de servidor único e servidor infinito, este modelo captura a simultaneidade das operações do dispositivo e as dependências sequenciais exigidas pelo servidor, oferecendo *insights* sobre a escalabilidade e robustez de soluções de FL.

3.3. Validação do Modelo SPN Comparando com Experimentos Reais

Foi realizado um experimento em um *Testbed* composto por 4 dispositivos Raspberry Pi 5 como clientes e um notebook como servidor, conectados por uma rede WiFi 802.11n para validar o modelo de SPN proposto. O experimento se inicia com o servidor definindo os pesos inicias do modelo que será treinado e selecionando um subconjunto de clientes para o treinamento. Essa amostragem é realizada de maneira uniforme, considerando 50% dos clientes disponíveis. Para garantir uma distribuição heterogênea de dados entre os clientes, um particionamento do *dataset* CIFAR10 foi realizado utilizando uma distribuição de *Dirichilet* com $alpha = 0.1$. Em seguida, cada cliente (i.e., dispositivo *raspberry*) treina uma modelo de rede convolucional com com 3 camadas convolucionais, cada uma com 32, 64 e 128 filtros, seguidas por uma MLP com 4 camadas densas com 1000 neurônios. Cada camada usa a função de ativação ReLu, e a saída do modelo é uma ativação *SoftMax* para prever as classes. Por fim, o treinamento do modelo foi utilizado uma função de perda de *Categorical Cross-Entropy*. Assim, após o treinamento do modelo, cada cliente reporta os gradientes para o servidor. Posteriormente, o servidor agrega os modelos recebidos utilizando o algoritmo de *Federated Averagin* (FedAvg) realizando a média ponderada dos modelos em relação tamanho do *dataset* dos clientes. Em seguida, o servidor seleciona novamente os clientes para dar continuidade ao treinamento federado. Esse processo se repete por 100 rodadas de comunicação.

Algumas simplificações foram adotadas para o experimento de validação: Não consideramos falhas, não consideramos disparos por *timeout* e apenas cohort=4 (o total de clientes disponíveis). Baseado no experimento, os tempos para os estágios da rodada de FL obtidos foram: T_ARRIVAL (0.25 s), T_AGGREG (2.928 s), T_TRAIN (1.017 s), T_SEND + T_RETURN (59.240 s), T_SELECT (0.480 s), T_COMPUTE (0.00008 s) e T_SAVE (0.00001 s). Aplicamos o T-Test de duas amostras para comparar o tempo

médio por rodada (T_{mean}) obtido no experimento e no modelo, tal como apresentado na Tabela 4. Para verificar a significância do T-Test, observamos que o valor p é superior a 0,05. Os resultados sugerem que não há diferença estatisticamente significativa entre as médias para um nível de significância comum $\alpha = 0.05$. Tal resultado ratifica a utilização do modelo SPN para o planejamento da arquitetura proposta deste trabalho, bem como adaptações e reutilizações à critério do avaliador.

Tabela 4. Resultados do Teste t.

Estatística	Média ± SD	t	p-valor
Modelo	62.80 ± 5.07	-0.29	0.77
Experimento	63.51 ± 7.65		

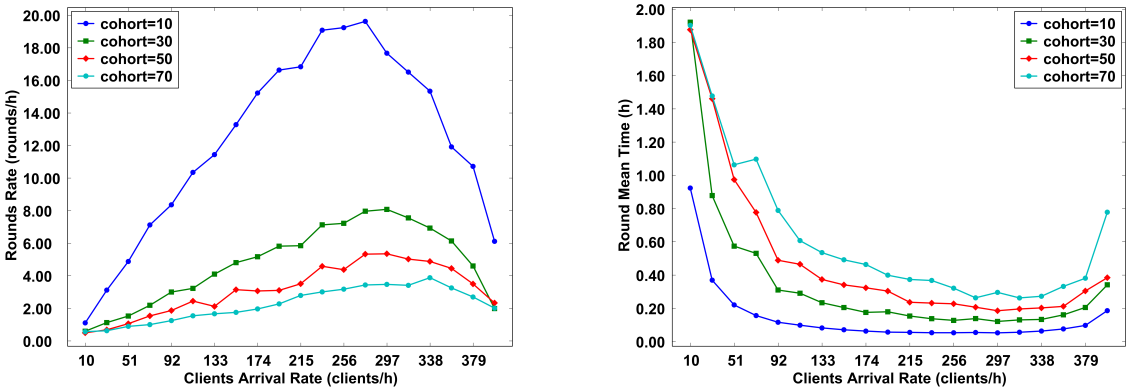
3.4. Estudo de Caso Utilizando o Modelo SPN Validado

Com o modelo validado podemos fazer estudos de caso com uma maior confiabilidade, onde o modelo de SPN foi então executado usando a ferramenta Mercury [Silva et al. 2015]. Esta seção apresenta uma análise detalhada do desempenho de sistemas de FL sob parâmetros variados. O método de avaliação utilizado é chamada de análise estacionária, a qual é uma técnica utilizada para estudar o comportamento de sistemas dinâmicos em estado de equilíbrio, ou seja, quando as condições do sistema não mudam mais ao longo do tempo. A avaliações consideram uma taxa de chegada de clientes (medida em clientes/hora) que varia de 10 clientes/hora a 400 clientes/hora e abrange cenários de baixa a alta disponibilidade de clientes. Estes valores são arbitrários, e o avaliador pode inserir quaisquer quantidades para atender suas necessidades de previsão. Os tempos alocados aos estágios da rodada de FL foram: T_AGGREG (0.00266 h), T_TRAIN (0.00169 h), T_SAVE (0.00048 h), T_SELECT (0.00047 h), T_SEND (0.00046 h), T_RETURN (0.00045 h) e T_COMPUTE (0.00043 h). Esses parâmetros foram estimados com base nos resultados do artigo [Bonawitz 2019]. Foram analisados os resultados para a taxa de rodadas (rodadas/hora) ou o tempo médio de rodada (horas) em diferentes variando o tamanho do coorte, configurações de tempo limite e probabilidades de falha.

O tamanho da coorte (cohort) determina o número de clientes que participam de uma única rodada de FL. Especificamente, coortes menores envolvem menos participantes, reduzindo a sobrecarga, mas potencialmente limitando a diversidade do modelo. Coortes maiores aumentam a participação, mas também introduzem atrasos e desafios de coordenação. A Figura 3 apresenta os resultados para a taxa de rodadas (rodadas/hora) ou o tempo médio de rodada (horas) variando o tamanho do coorte. Visando que os outros parâmetros não influenciassem esta análise, a probabilidade de falha e o tempo limite foram fixados com 10% (baixa probabilidade) e 1,0 hora (tempo alto), respectivamente. Ao analisar os resultados da Figura 3a é possível observar que a taxa de rodadas aumenta com a taxa de chegada do cliente para todos os tamanhos de coorte, atingindo o pico antes de estabilizar ou declinar devido à saturação do sistema. Coortes menores (por exemplo, 10 clientes) atingem a maior taxa de rodadas, com um máximo de aproximadamente 19 rodadas/hora a 250 clientes/hora. Coortes maiores (por exemplo, 70 clientes) resultam em uma taxa de rodadas reduzida de cerca de 2,5 rodadas/hora na mesma taxa de chegada de clientes. Esses resultados sugerem que coortes menores são ideais quando atualizações rápidas são críticas, como no monitoramento em tempo real.

A Figura 3b apresenta o tempo médio de rodada variando o tamanho da coorte. O tempo médio de rodada reflete inversamente a taxa de rodadas, com coortes menores

concluindo rodadas mais rápido. Para 10 clientes, o tempo médio de rodada é tão baixo quanto 0,054 horas a uma taxa de chegada de clientes de 300 clientes/hora, enquanto coortes maiores (70 clientes) requerem aproximadamente 0,36 horas por rodada. Esses resultados enfatizam que coortes maiores são mais adequadas para aplicativos que priorizam a diversidade de modelos em vez do tempo de resposta.



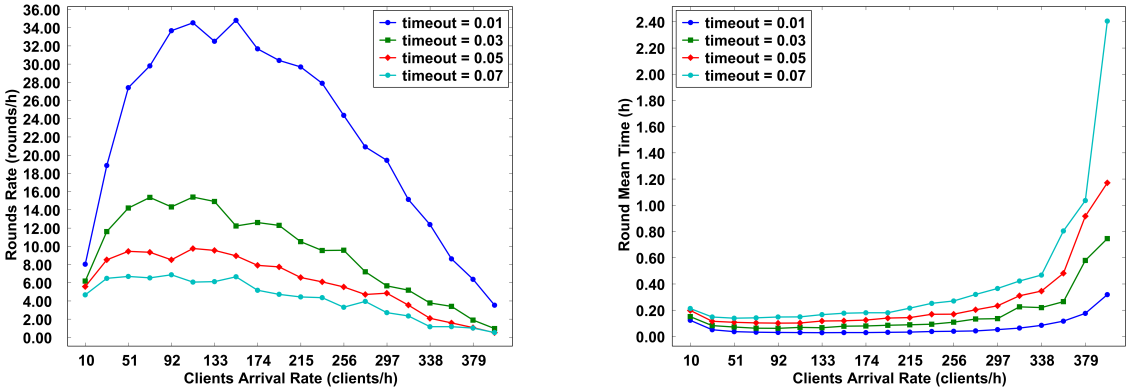
(a) Rounds Rate - cohort variation. **(b) Mean Round Time - cohort variation.**

Figura 3. Análise considerando a variação de cohort.

As configurações de tempo limite determinam quanto tempo o sistema espera por atualizações do cliente antes de prosseguir para a próxima rodada. Tempos limite mais curtos priorizam rodadas mais rápidas, mas podem excluir clientes mais lentos. Tempos limite mais longos aumentam a chance de incluir mais participantes, mas podem introduzir atrasos. A Figura 4 apresenta os resultados para a taxa de rodadas (rodadas/hora) ou o tempo médio de rodada (horas) variando as configurações de tempo limite. Visando que os outros parâmetros não influenciem esta análise, a probabilidade de falha e o coorte foram fixados com 10% (baixa probabilidade) e 3000 clientes (número inalcançável), respectivamente. Ao analisar os resultados da Figura 4a é possível observar que a taxa de rodadas atinge o pico em configurações de tempo limite mais curtas, como 0,01 horas, atingindo um máximo de aproximadamente 34 rodadas/hora a uma taxa de chegada de clientes de 100 clientes/hora. À medida que os valores de tempo limite aumentam (por exemplo, 0,03 horas), a taxa de rodadas cai para cerca de 14 rodadas/hora, pois mais tempo é gasto esperando por respostas do cliente. Isso sugere que as configurações de tempo limite intermediárias são ideais para equilibrar a taxa de transferência e a inclusão do cliente, especialmente em redes com tempos de resposta de dispositivos variáveis.

A Figura 4b apresenta o tempo médio de rodada variando o tempo limite. Tempos limite mais curtos resultam em tempos médios de rodada mais baixos. Considerando 400 clientes/h, um timeout de 0,01 hora produz um tempo médio de rodada de aproximadamente 0,3 hora. Por outro lado, timeouts mais longos, como 0,07 hora, aumentam o tempo médio de rodada para aproximadamente 2,4 horas. Essas descobertas destacam a importância de ajustar as configurações de timeout com base nas necessidades do aplicativo — por exemplo, redes industriais de IoT podem se beneficiar de timeouts intermediários para equilibrar velocidade e inclusão.

A Figura 5 apresenta os resultados para a taxa de rodadas (rodadas/hora) ou o tempo médio de rodada (horas) variando a probabilidades de falha. A probabilidade de falha representa a probabilidade de um cliente não concluir seu treinamento durante uma



(a) Rounds Rate - timeout variation.

(b) Mean Round Time - timeout variation.

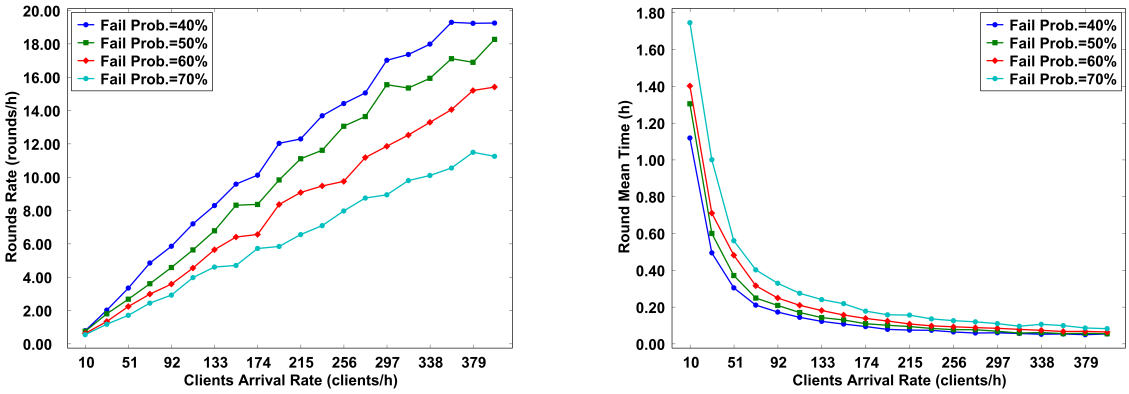
Figura 4. Análise considerando a variação de timeout.

rodada. Nos gráficos analisados, as probabilidades de falha variam de 40% a 70%, onde probabilidades mais altas levam à redução da taxa de rodadas e ao aumento de atrasos devido a novas tentativas ou atualizações incompletas. Ao observar os resultados da Figura 5a é possível concluir que o tempo limite e a coorte foram fixados com 1,0 hora (tempo alto) e 10 clientes (número pequeno), respectivamente. Neste caso, a agregação será acionada por meio da estratégia de coorte. A taxa de rodadas diminui conforme a probabilidade de falha aumenta. Com uma probabilidade de falha de 40%, a taxa de rodadas atinge o pico em aproximadamente 19 rodadas/hora. Com 60% de probabilidade de falha, ela cai para cerca de 15 rodadas/hora, demonstrando a importância de minimizar as taxas de falha para manter alto rendimento. Por exemplo, no gerenciamento de frotas de drones, garantir conectividade robusta pode ajudar a reduzir falhas e sustentar o desempenho do sistema.

A Figura 5b apresenta a probabilidade de falha média variável no tempo da rodada. Maiores probabilidades de falha aumentam o tempo médio da rodada. Com 10 clientes/hora e 40% de probabilidade de falha, o tempo médio da rodada é de aproximadamente 1,18 horas. Conforme a probabilidade de falha aumenta para 70%, o tempo médio da rodada excede 1,8 horas. Esses resultados enfatizam o valor de projetar sistemas tolerantes a falhas, particularmente para aplicações de missão crítica, como coordenação de veículos autônomos, onde atrasos podem impactar significativamente as operações.

4. Conclusão

Este trabalho apresenta uma abordagem para modelar e avaliar o desempenho e a escalabilidade de sistemas de FL sob várias condições operacionais o que não é tarefa trivial em ambientes simulados, emulados e reais. Os resultados demonstram como diferentes parâmetros afetam a taxa de rodada e o tempo médio de rodada. Tamanhos menores de coorte e tempos limite mais curtos levaram a taxas de rodada mais altas, tornando-os apropriados para aplicativos que exigem atualizações frequentes. Em contraste, coortes maiores e tempos limite mais longos facilitaram a inclusão de mais clientes e garantiram uma representação de dados mais ampla. A análise das probabilidades de falha destacou a importância da confiabilidade do cliente para manter o desempenho do sistema. Essas descobertas fornecem insights acionáveis para otimizar implantações de FL em vários domínios, como IoT industrial e sistemas de saúde. O trabalho futuro se concentrará



(a) Rounds Rate - failure probability variation.

(b) Mean Round Time - failure probability variation.

Figura 5. Análise considerando a variação de probabilidade de falha.

na aplicação do modelo SPN proposto a contextos específicos. Na IoT industrial, por exemplo, o modelo pode ser usado para avaliar o impacto da heterogeneidade do dispositivo e da variabilidade da rede no desempenho do FL, ajudando a otimizar estratégias de implantação sistemas IoT em larga escala.

Agradecimentos

Este trabalho foi parcialmente financiado pelo CNPq, processo #405940/2022-0, e CAPES, processo #88887.954253/2024-00.

Referências

Albelaihi, R., Sun, X., Craft, W. D., Yu, L., and Wang, C. (2021). Adaptive participant selection in heterogeneous federated learning. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE.

Banabilah, S., Aloqaily, M., Alsayed, E., Malik, N., and Jararweh, Y. (2022). Federated learning review: Fundamentals, enabling technologies, and future applications. *Information processing & management*, 59(6):103061.

Beutel, A., Topcuoglu, E., Ozisik, C., and Steiger, E. (2020). Flower: A friendly federated learning research framework. In *Systems and Machine Learning Conference*.

Bonawitz, K. (2019). Towards federated learning at scale: Syste m design. *arXiv preprint arXiv:1902.01046*.

DeGlopper, D. R. (1992). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modeling. by raj jain. new york: John wiley and sons, 1991. pp. 720. (hardcover). *International Journal of Legal Information*, 20(1):63–64.

Du, M., Zheng, H., Feng, X., Chen, Y., and Zhao, T. (2022). Decentralized federated learning with markov chain based consensus for industrial iot networks. *IEEE Transactions on Industrial Informatics*, 19(4):6006–6015.

Gilmer, M., Sohl-Dickstein, J., Schoenholz, S. S., Bauer, M. S., and Gilmer, J. (2021). Fedjax: A scalable jax framework for federated learning. *arXiv preprint arXiv:2108.02117*.

Joshi, A., Agarwal, A., Agarwal, D., Murthy, B., Chaudhary, M., Agarwal, S., Sekar, A., Choudhary, D., and Girdhar, R. (2021). Flute: A scalable federated learning simulator. *arXiv preprint arXiv:2110.06203*.

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.

Kholod, I., Yanaki, E., Fomichev, D., Shalugin, E., Novikova, E., Filippov, E., and Nordlund, M. (2020). Open-source federated learning frameworks for iot: A comparative review and analysis. *Sensors*, 21(1):167.

Kim, M. G., De M Bastos, C. P. M. J., Park, J., Hajiesmaili, M., Oh, S., Ma, M., and Kim, M. (2021). Flsim: Scalable and extensible federated learning simulator. *arXiv preprint arXiv:2107.03309*.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., and He, B. (2021). A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3347–3366.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60.

Liu, T., Wang, H., and Ma, M. (2024). Federated learning with efficient aggregation via markov decision process in edge networks. *Mathematics*, 12(6):920.

Maciel, P. R. M. (2023). *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. Chapman and Hall/CRC.

Pu, J., Fu, X., Dong, H., Zhang, P., and Liu, L. (2024). Dynamic adaptive federated learning on local long-tailed data. *IEEE Transactions on Services Computing*.

Savazzi, S., Nicoli, M., Bennis, M., Kianoush, S., and Barbieri, L. (2021). Opportunities of federated learning in connected, cooperative, and automated industrial systems. *IEEE Communications Magazine*, 59(2):16–21.

Silva, B., Matos, R., Callou, G., Figueiredo, J., Oliveira, D., Ferreira, J., Dantas, J., Lobo, A., Alves, V., and Maciel, P. (2015). Mercury: An integrated environment for performance and dependability evaluation of general systems. In *45th dependable systems and networks conference (DSN)*, pages 1–4.

TensorFlow Federated (2021). TensorFlow Federated: Machine Learning on Decentralized Data.

Turgay, S. (2022). Blockchain management and federated learning adaptation on healthcare management system. *International Journal of Intelligent Systems and Applications*, 14(5):1.

Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., and Zhang, W. (2023). A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2):513–535.

Wilhelmi, F., Giupponi, L., and Dini, P. (2021). Blockchain-enabled server-less federated learning. *arXiv preprint arXiv:2112.07938*, pages 1–14.