Inferência da qualidade do serviço em enlaces de rede através de um método baseado em aprendizado federado

Nicolas A. Alves da Silva[1], **Carlos A. Vieira Campos**[1], **Sidney C. de Lucena**[1]

[1]Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Av. Pasteur, 458, Urca – 22290-255 – Rio de Janeiro – RJ – Brasil

nicolasaasilva@edu.unirio.br, {beto,sidney}@uniriotec.br

Abstract. *Due to the growing demand for Internet access, the request for improvements in the infrastructure of Internet Service Providers and networks of mobile access points also grows. This new scenario motivates the adoption of machine learning techniques for monitoring and managing networks. This work proposes a method that uses federated learning to infer the quality of service in the network. The proposed model uses, as input, traffic matrices made of measurements collected at the source nodes themselves. By using homomorphic cryptography, the source nodes maintain the privacy of their traffic measurement data, preventing them from information leakage attacks during the training process. Results of experiments indicate that the proposed method outperformed centralized machine learning techniques, with a smaller number of rounds until model convergence.*

Resumo. *A crescente demanda de acesso à Internet acarreta no aumento da demanda por melhorias nas infraestruturas das redes provedoras de Internet e redes de pontos de acesso móvel, e a complexidade deste novo cenário motiva o uso de técnicas de aprendizado de máquina para monitoramento e gerenciamento de redes. Este trabalho propõe um método baseado em aprendizado federado para inferir a qualidade de serviço na rede. Como entrada deste modelo, são usadas matrizes de tráfego contendo medidas coletadas nos próprios nós de origem. Utilizando criptografia homomórfica, os nós de origem mantêm a privacidade dos seus dados de medição, prevenindo-os de ataques de vazamento de informação durante o processo de treinamento. Resultados dos experimentos realizados apontam que o método proposto superou o aprendizado de máquina centralizado em termos de desempenho, e com um menor número de rodadas até a convergência do modelo.*

1. Introdução

Ao longo dos últimos anos, vem ocorrendo uma grande popularização e aumento da demanda por uso da Internet. Segundo a Agência Nacional de Telecomunicações (Anatel), a quantidade dos acessos de Banda Larga saltou de 26,9 milhões em 2017 para 44,4 milhões no fim de 2022[1], e neste mesmo período, o número de acessos com mais de 34 Mbps saltou de 2,9 milhões para 38,3 milhões. Estes dados indicam uma evolução da relação entre quantidade e velocidade dos acessos.

[1]http://www.anatel.gov.br/paineis/acessos/banda-larga-fixa

Tal crescimento também se estende ao cenário mundial de provedores de acesso à Internet (*Internet Service Providers* – ISPs) e ao contexto de dispositivos móveis. De acordo com [Clausen et al. 2022], o consumo de dados no Reino Unido, por exemplo, cresceu mais que o dobro nos últimos dois anos, a maior parte deste conteúdo relacionado ao *streaming* de TV. Quanto ao contexto de dispositivos móveis, o aumento de demanda é percebido pela própria evolução das gerações de redes móveis. Os autores de [Navarro-Ortiz et al. 2020] mostram que o padrão IMT-2020, da *International Mobile Telecommunications*, para redes 5G pode alcançar picos de velocidade de 10 Gbps e garante ao usuário uma taxa de transferência de dados de 100 Mbps para *download*.

Diante deste aumento de demanda, fica evidente a necessidade de um melhor monitoramento da qualidade de serviço (QoS) e de um gerenciamento mais eficaz das redes de ISPs ou de pontos de acesso a dispositivos móveis. Uma medida de desempenho importante para isso é a matriz de tráfego entre seus nós, que contém medidas do tráfego de origem e destino entre todos os nós da rede, num dado instante de tempo. Se uma coleta de matrizes de tráfego possuir uma taxa de amostragem constante, então elas podem ser usadas para gerar estimativas de qualidade de serviço na rede [Uhlig et al. 2006]. A busca por formas menos custosas de obter matrizes de tráfego com alta periodicidade gerou trabalhos que fazem uso de aprendizado de máquina com este fim [Mesquita and Assis 2019, Amoroso et al. 2020]. No entanto, como os valores contidos nas matrizes de tráfego revelam muito do perfil dos clientes da rede, técnicas centralizadas de aprendizado de máquina tornam esses dados vulneráveis. Além disso, o paradigma centralizado gera alto custo ao servidor que concentra todas tarefas do processo.

Este artigo propõe um método que objetiva medir o tráfego dos nós da rede, organizar estas medidas no formato de matriz de tráfego e inferir a QoS na rede a partir desta matriz. Para isso, propõe-se uma solução baseada em aprendizado federado (*Federated Learning* - FL) para treinar o modelo de predição de QoS. Segundo [McMahan et al. 2017, Lo et al. 2021], o FL é uma abordagem na qual o treinamento do modelo é realizado de maneira distribuída. O sistema de medição deste método e o armazenamento das matrizes de tráfego usadas no treinamento do modelo também serão distribuídos. Para manter a privacidade dos usuários da rede, as matrizes de tráfego serão formadas por medidas usando criptografia homomórfica. De acordo com [Aono et al. 2017], além de manter a privacidade dos dados brutos dos usuários, isso previne o vazamento das informações contidas nas contribuições locais, transmitidas durante o treinamento distribuído. Portanto, as contribuições do presente artigo são as seguintes: (1) a proposição de um método baseado em aprendizado federado, usando criptografia homomórfica, para inferir a QoS na rede, e (2) a apresentação de um estudo de caso com um experimento que utiliza dados reais.

As próximas seções deste artigo estão organizadas da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve os conceitos preliminares do método; a Seção 4 apresenta o método de aprendizado para o modelo proposto de predição de QoS; a Seção 5 descreve a elaboração de um experimento e a análise de seus resultados; por fim, a Seção 6 apresenta as conclusões do trabalho.

2. Trabalhos relacionados

Soluções para problemas de otimização de recursos de rede e para balanceamento de carga dentro da infraestrutura de redes são temas da investigação de alguns trabalhos que serão discutidos ao longo desta seção. Os trabalhos [Tu et al. 2015, Monteiro et al. 2019, She et al. 2022] serão explorados enquanto diferentes contrapontos a esta proposta. Como metodologia, o desenvolvimento de [Amoroso et al. 2020] será um importante referencial para este trabalho, conforme apresentado ao final desta seção, e [Aono et al. 2017] destaca a importância de proteção de dados em Aprendizados de Máquina.

Em [Tu et al. 2015], encontra-se a descrição de um método de otimização de recursos para melhorar a eficiência de aplicações que transferem dados entre cliente e servidor. Nele, os autores combinam a tecnologia de SDNs e de redes Clos para diminuir a latência de destas aplicações. A topologia de rede Clos descrita neste trabalho apresenta um caminho linear entre os clientes e o servidor. Este caminho linear possui divisões em alguns estágios, cada um com diferentes *switches* em paralelo. Um *switch* do estágio inicial transfere dados para um *switch* do próximo estágio; este novo *switch*, por sua vez, transfere dados para um *switch* do estágio posterior; e assim sucessivamente. Com o auxílio de um controlador SDN, um algoritmo determinístico é aplicado para encontrar caminho ótimo do cliente para o servidor, através das escolhas de *switches* nas mudanças de estágios neste caminho. Estas escolhas de *switches* levam em consideração o *delay* de transmissão de dados e a frequência de utilização dos recursos da rede. O método proposto está muito associado ao caso particular de redes com topologia Clos e usando a abordagem SDN. Por isso, os autores adotaram uma técnica que utiliza o controlador da SDN para treinar um modelo de forma centralizada e a otimização de recursos desejada não é abrangente o suficiente para atingir outros tipos de topologia de redes.

Em [Monteiro et al. 2019], os autores propõem uma abordagem para solucionar um problema de otimização e realocação de recursos RMLSA (*Routing, Modulation Level and Spectrum Assignment*) em redes ópticas elásticas. Tais redes se diferenciam de redes ópticas convencionais pela possibilidade de dividir o espectro óptico dos seus canais em intervalos de frequência chamados de *slots*. Portanto, o problema de RMLS é solucionado através dos algoritmos *Guard Band according to Use of the Network* (GBUN), *Complete Sharing* e QoT (*Quality of Transmission*). O algoritmo GBUN mede a utilização do espectro da rede e o algoritmo *Complete Sharing* seleciona uma rota de acordo com os *slots* de frequência menos utilizados e o algoritmo QoT seleciona um formato de modulação. O trabalho apresenta uma importante solução de problemas relacionados a otimizações e alocação de recursos em redes, porém ele também descreveu uma forma de treinamento centralizada para seu modelo. Vale a pena ressaltar que as ações aprendidas pelo modelo, a fim de otimizar os recursos da rede, fazem sentido apenas em redes com enlaces óticos.

Os autores de [She et al. 2022] propõem um método de balanceamento de carga em redes baseado em *Blockchain*. Esse modelo de *Blockchain* é utilizado para armazenar os dados de medições dos enlaces da rede de forma distribuída. Esses dados foram coletados através de uma adaptação da telemetria *in-band* (INT) para associar a carga do próprio controlador da rede com as medidas de *status* de cada domínio de rede. Por fim, os autores propõem um método baseado em *Deep Reinforcement Learning* que usa a carga do controlador global e os *status* dos domínios de rede para selecionar *switches* de domínios de rede sobrecarregados e, com isso, atingir um balanceamento de carga global.

Apesar dos aspectos distribuídos que envolvem *Blockchains*, a solução apresentada possui um *dataset* que será copiado para todos os nós da rede. Esta é uma característica clássica de *Blockchains* para garantir a segurança de sua cadeia. Porém, em grandes *datasets*, a carga da rede utilizada em armazenamento aumentará bruscamente. Além disso, os processamentos de operações em *Blockchains* são realizados a partir da seleção aleatória de um conjunto de nós da rede. Isso gera momentos de ociosidade para os nós dessa rede.

Em [Amoroso et al. 2020], os autores descrevem uma técnica de aprendizado de máquina a partir de matrizes de tráfego. Baseada no problema de inferir um aumento de resolução em imagens, essa técnica tem o objetivo de inferir uma matriz de tráfego completa (alta resolução), a partir de uma versão dela com redução de dimensionalidade (baixa resolução). Para tal objetivo, os autores treinaram um modelo supervisionado usando um *dataset* que contém matrizes de tráfego com redução de dimensionalidade e suas correspondentes matrizes completas de tráfego. Além disso, esse trabalho demonstra que o aprendizado apresentou uma convergência do modelo mais rápida usando a abordagem federada. A técnica proposta pode ser utilizada com inúmeras finalidades, e.g. para reduzir o custo de servidores para um armazenamento de séries históricas de matrizes de tráfego. Entretanto, a aplicação desta técnica poderia envolver o aprendizado de matrizes de tráfego com resolução completa, porém, com medidas criptografadas. Dessa forma, um dispositivo atacante ou um usuário da rede honesto e curioso não seria capaz de descobrir as medidas de tráfego dos demais nós da rede, a partir do modelo global treinado e uma matriz de tráfego com redução de dimensionalidade.

Em [Aono et al. 2017], os autores apresentam um sistema para manter a privacidade dos dados locais presentes em dispositivos participantes de um treinamento global de aprendizado de máquina. Primeiramente, demonstram que, usando o método de Gradiente Descendente Estocástico (*Stochastic Gradient Descent* – SGD), uma rede neural pode ser treinada através da participação dos gradientes de diferentes dispositivos. Então, descrevem o conceito de vazamento da informação de gradientes em treinamentos usando SGD e como esse vazamento pode revelar os dados locais dos participantes. Finalmente, os autores apresentam uma solução, usando criptografia homomórfica, para manter a privacidade dos dados locais em vazamentos da informação de gradientes num aprendizado SGD. O método proposto apresenta uma forma interessante para manter a privacidade dos dados locais em técnicas de aprendizado baseadas em FL. Isto porque abordagens FL utilizam métodos SGD. Dessa forma, é possível aplicar criptografia homomórfica em dados locais de aprendizados que utilizam a abordagem federada.

Os trabalhos [Tu et al. 2015, Monteiro et al. 2019, She et al. 2022], que foram citados acima, são relacionados ao problema de otimização e realocação de recursos. Logo, suas finalidades se assemelham com a deste artigo, que busca inferir a QoS na rede para tornar possível o seu balanceamento de carga. Porém, uma vez que esses trabalhos usam aprendizado centralizado, eles concentram, no servidor, o processamento e o armazenamento dos dados referente a todos os nós da rede.

3. Conhecimentos preliminares

Estão seção apresenta os conceitos nos quais o método proposto neste artigo se baseou: o de gradiente descendente e o de aprendizado federado.

3.1. Gradiente Descendente

De acordo com [Hartmann 2018], no aprendizado supervisionado, uma função de perda $p(\mathbf{X}^{(t)})$ quantifica o quão perto uma previsão $f(\mathbf{X}^{(t)})$ de um modelo está da resposta correta $\mathbf{Y}^{(t)}$. Os parâmetros do modelo devem ser escolhidos para minimizar a perda. Estes parâmetros são os pesos $w = \{w_0, w_1, w_2, \ldots, w_j\}$ da função preditiva $f(\mathbf{X}^{(t)})$ do modelo, a qual utiliza uma função de ativação q, por exemplo, a função logística: $f(\mathbf{X}^{(t)}) = q(w_0\mathbf{x}_0^{(t)} + w_1\mathbf{x}_1^{(t)} + w_2\mathbf{x}_2^{(t)} + \cdots + w_j\mathbf{x}_j^{(t)})$. Então, uma função de perda é escolhida para ser minimizada através dos dados no *dataset* separados para treinamento. Segundo [Hartmann 2018], uma escolha popular para essa função de perda é o erro quadrático: $p(\mathbf{X}^{(t)}) = (\ f(\mathbf{X}^{(t)}) - \mathbf{Y}^{(t)}\ )^2$.

Na literatura, há diferentes algoritmos capazes de minimizar $p(\mathbf{X}^{(t)})$. De acordo com [Goodfellow et al. 2016] e [Bishop and Nasrabadi 2006], um desses algoritmos é o Gradiente Descendente. Trata-se de um algoritmo iterativo em que os pesos do modelo são movidos repetidamente para uma direção que otimiza a função do modelo $f(\mathbf{X}^{(t)})$, através da minimização da função de perda $p(\mathbf{X}^{(t)})$. Portanto, o algoritmo Gradiente Descendente deve ser inicializado com um conjunto arbitrário de pesos $w = \{w_0, w_1, w_2, \ldots, w_j\}$ e as funções $f(\mathbf{X}^{(t)})$ e $p(\mathbf{X}^{(t)})$, que devem ser diferenciáveis.

Em seguida, começa o processo de otimização iterativa. Em cada rodada de iteração i, são calculadas as derivadas parciais da função de perda em relação aos pesos do modelo. Um fator de escala η, chamado taxa de aprendizado, é aplicado às derivadas parciais. Cada parâmetro w_j é, então, atualizado repetidamente usando a seguinte equação: $w_j^{(i)} = w_j^{(i-1)} - \eta \times \left(\frac{\partial p}{\partial w_j}\right)$.

O algoritmo Gradiente Descendente termina quando a atualização, mediante a equação acima, converge em valores praticamente iguais de acordo com uma tolerância definida. Com este valores finais dos pesos do modelo, o aprendizado pode ser avaliado através do conjunto de dados separado para teste. A função f do modelo passa a ser composta com os pesos $w = \{w_0, w_1, w_2, \ldots, w_j\}$ e, através da comparação de $f(\mathbf{X}^{(t)})$ e $\mathbf{Y}^{(t)}$, o aprendizado do modelo é avaliado.

3.2. Aprendizado federado

De acordo com [Thonglek et al. 2020, McMahan et al. 2016], a abordagem federada de aprendizado de máquina treina um modelo global em um servidor centralizado usando processamento de modelos locais em diferentes dispositivos. A aplicação do aprendizado federado tem, como vantagens, a utilização dos recursos de processamento dos dispositivos locais e a falta da necessidade de transferência de seus dados brutos para um servidor.

Dentre outros algoritmos de aprendizado federado, existem dois que são básicos: *Federated Stochastic Gradient Descent* (FedSGD) e *Federated Averaging* (FedAVG). Ambos os algoritmos federados são baseados no aprendizado de modelos que utilizam Gradiente Descendente.

No FedSGD, o modelo global e os modelos locais são treinados da seguinte maneira: em cada rodada de iteração, o servidor transmite o modelo global atual com os parâmetros $w^{(i)}$ para todos os clientes (i é a iteração atual). Cada cliente k, então, calcula o seu vetor gradiente $g_k = (\frac{\partial p}{\partial w_0}, \frac{\partial p}{\partial w_1}, \ldots, \frac{\partial p}{\partial w_j})$ usando seus dados de treinamento locais e envia o gradiente calculado para o servidor. O servidor calcula a média dos gradientes

recebidos de todos os clientes e gera o novo modelo global com parâmetros $w^{(i+1)}$, de acordo com a equação $w^{(i+1)} = w^{(i)} - \eta \times \sum_{k=1}^{K} g_k^{(i)}$

No algoritmo FedAVG, de acordo com [McMahan et al. 2016], as equações de atualização dos modelos, em cada iteração, são diferentes em relação ao FedSGD, tanto para o modelo global no servidor e quanto para os modelos locais nos dispositivos cliente. Em FedAVG, os dispositivos calculam os seus pesos locais através dos seus dados, de acordo com a equação $w^{(i+1)} = w^{(i)} - \eta \times g_k^{(i)}$, e o servidor agrega os pesos de todos os dispositivos para atualizar os parâmetros do modelo global: $w^{(i+1)} = \sum_{k=1}^{K} w_k^{(i+1)}$.

Ao fim de cada iteração, o servidor transmite o novo modelo $w^{(i+1)}$ para os clientes. Cada um deles recebe o modelo global mais recente e, novamente, são atualizados os modelos locais $w_k^{(i+2)}$, que serão enviados ao servidor para finalizar mais uma iteração.

4. Método proposto para a predição de QoS na rede

Nesta seção, descreveremos o nosso método, proposto com o objetivo de inferir a QoS nos enlaces de uma rede, a partir de matrizes de tráfego referentes aos seus nós. Estas matrizes contêm, em seus valores, as medidas de tráfego com origem e destino entre todos os nós da rede. Para alcançar este objetivo, o método será dividido em três etapas: coleta de dados, pré-processamento e treinamento do modelo. Cada etapa é composta da realização de algumas tarefas. A Figura 1 ilustra todas as etapas do método e resume as tarefas de cada uma delas.

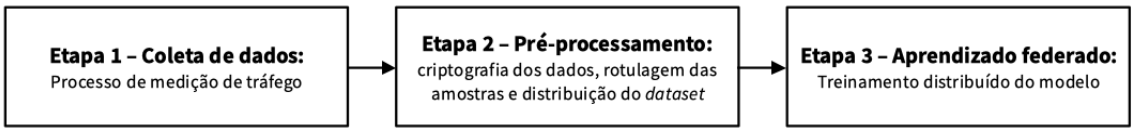

Figura 1. Etapas do método proposto.

Nas subseções seguintes, serão descritas as características e detalhes das tarefas referentes a cada etapa do método proposto.

4.1. Etapa 1: Coleta de dados

Esta etapa do método compreende duas tarefas. A primeira delas se trata de um sistema de medição de tráfego a partir de cada nó da rede. Esta medição é realizada de uma forma distribuída, na qual cada nó da rede mede seus próprios tráfegos com destino a todos os nós. Os dados de medição de tráfego coletados nesta etapa serão utilizados na próxima etapa para a criação das amostras de matrizes de tráfego. Logo, as medidas de tráfego, coletadas de forma distribuída nesta etapa, devem ser síncronas e periódicas, com tempo de amostragem fixo, ao comando de um servidor central da rede. A Figura 2 ilustra este sistema de medição.

Como é possível perceber pela Figura 2, o valor $x_{AB}^{(t)}$ representa uma medida de tráfego com origem no nó A e destino no nó B, no instante de tempo t . Em um mesmo instante de tempo t, o nó A também deve reunir medidas de tráfego com destino a todos os nós da rede. Então, no nó A, o vetor $x_A^{(t)}$ contém todas as suas medidas referentes ao instante de tempo t, i.e. o vetor $x_A^{(t)} = (x_{AA}^{(t)}, x_{AB}^{(t)}, x_{AC}^{(t)})$.

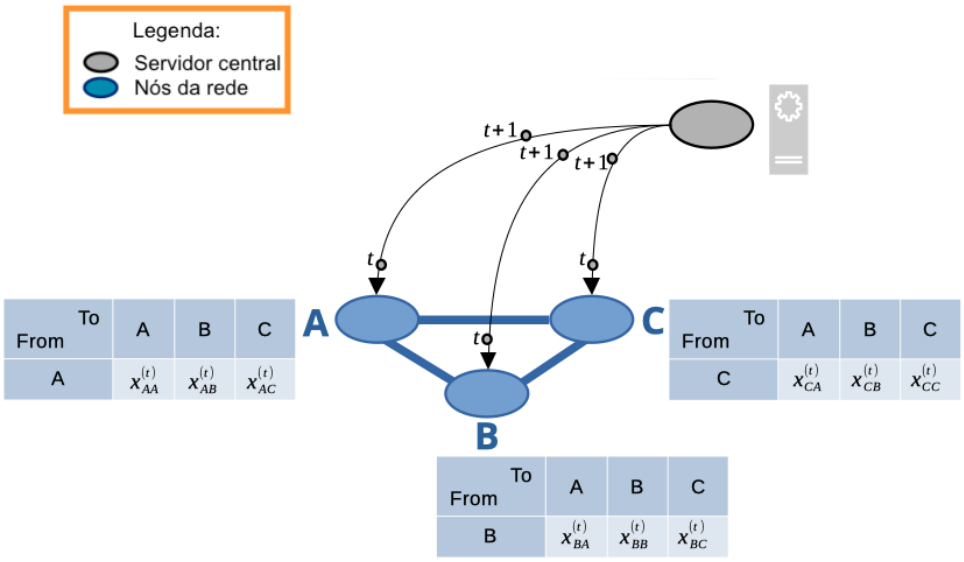

Figura 2. Medição de tráfego realizada de forma distribuída, síncrona e periódica. Nós da rede A, B e C, em azul, e servidor central, em cinza, para sincronização do tempo de amostragem.

Esta mesma medição deve ser realizada em todos os demais nós da rede. Se um vetor de medidas apresentar um elemento faltante, devido a uma falha de medição qualquer, então, ele deve ser descartado. Este vetor descartado não deve ser considerado para avançar à realização da próxima tarefa.

A segunda tarefa desta etapa é a medição do desempenho dos enlaces de rede. Esta tarefa também é realizada de forma distribuída e síncrona. Para esta etapa, assume-se que cada nó conhece as características da topologia da rede. Segundo uma escolha global de métrica de medição, cada nó mede o desempenho de enlace y, sentido por seus pacotes.

Existem várias métricas de desempenho de enlace de rede que podem ser consideradas para esta tarefa. Uma delas é a taxa de utilização dos enlaces de rede. A taxa de utilização de cada enlace de rede é calculada pela divisão entre a sua taxa de transferência e a sua capacidade de banda passante. Por isso, sendo esta a métrica escolhida para a avaliação dos enlaces de rede, é importante que cada nó da rede aplique o algoritmo de *Dijkstra* para descobrir por quais enlaces passam os tráfegos originados no nó. As medidas de tráfego de cada nó são as taxas de transferência com todos os nós da rede.

No exemplo da Figura 3, ao aplicar o algoritmo de *Dijkstra*, o caminho menos custoso entre os nós A e B, por hipótese, percorre os enlaces L2 e L3; enquanto entre A e C, apenas o enlace L2. As capacidades de banda nos enlaces L1, L2 e L3 da rede, são respectivamente bw_{L1}, bw_{L2} e bw_{L3}. Como cada nó possui apenas as taxas de transferência do seu tráfego de origem até todos os nós, então, para obter a taxa de utilização do enlace AC pelo tráfego originado em A, calcula-se a soma das taxas de transferência de tráfego entre A e B e entre A e C, i.e. $x_{AB} + x_{AC}$. Já a taxa de transferência do tráfego de A em L1 é zero e, em L3, é x_{AB}. Logo, os enlaces L1, L2 e L3 possuem as respectivas taxas de utilização em relação ao tráfego gerado por A: 0, $(x_{AB} + x_{AC})/bw_{L2}$ e x_{AB}/bw_{L3} .

Para reduzir o tempo de processamento da coleta das medidas de QoS, a aplicação

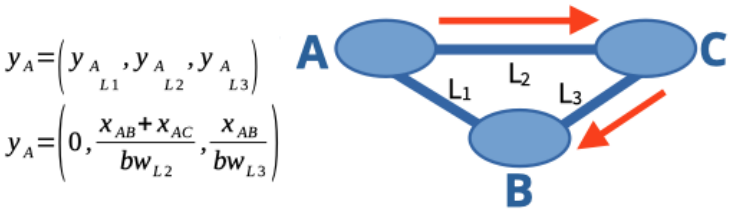

Figura 3. Medição de desempenho nos enlaces de rede (utilização de banda).

deste método deve considerar taxas de amostragem relativamente baixas, conforme a percepção da qualidade de experiência (QoE) dos usuários. Em [Grigoriou et al. 2018], por exemplo, os autores avaliam a QoS da rede e a QoE de uma aplicação de vídeo usando um monitoramento com tempo de amostragem da ordem de minutos.

4.2. Etapa 2: Pré-processamento

Esta etapa do método tem o objetivo de fornecer um *dataset* distribuído para a realização do treinamento federado e supervisionado, que será descrito na Subseção 4.3. Em paralelo, esta etapa também é responsável pela realização de operações que garantem a privacidade dos dados obtidos pela medição nos nós da rede. Para alcançar esses dois objetivos, durante esta etapa, os nós da rede realizam a sequência de tarefas listadas abaixo:

1. Cada nó aplica uma criptografia homomórfica nos valores dos seus vetores de medidas de tráfego: $x \rightarrow \mathbf{x}$.
2. A cada instante de tempo, um nó da rede é escolhido para receber todos os vetores com medidas criptografadas e todas as correspondentes medidas de desempenho de enlace.
3. Os nós da rede utilizam os vetores $\mathbf{x}$, do mesmo instante t, para formar uma amostra de matriz de tráfego criptografada: $\mathbf{X}^{(t)} = (\mathbf{x}_A^{(t)}, \mathbf{x}_B^{(t)}, \cdots)$.
4. Se ocorreu uma falha de medição na etapa anterior ou falha de transferência no item 2 desta etapa, haverá uma lacuna na matriz de tráfego criptografada. Neste caso, a matriz inteira deve ser descartada e desconsiderada para as tarefas futuras.
5. Cada nó agrega todas as medidas de desempenho nos enlaces de rede, referentes aos diferentes nós e ao mesmo instante de tempo. Considerando a escolha da métrica de desempenho de enlace como a taxa de utilização de banda, segue a agregação das medidas de L1 na rede da Figura 3: $y_{L1}^{(t)} = y_{A_{L1}}^{(t)} + y_{B_{L1}}^{(t)} + y_{C_{L1}}^{(t)}$.
6. De acordo com essas medidas de desempenho, são criados níveis de QoS. Segundo uma regra criada na implementação do método, este processo resulta no mapeamento de rótulos de QoS para cada enlace: $y \rightarrow \mathbf{y}$. Considerando todos os enlaces, segue a transformação dos desempenhos em rótulos: $Y \rightarrow \mathbf{Y}$.

Ao fim das tarefas citadas acima, cada nó possui um conjunto de amostras, em que cada uma delas se apresenta no formato de par de dados. Este par é composto das seguintes componentes: um vetor de medidas de tráfego criptografadas $\mathbf{X}$ e um vetor de taxa de utilização de banda nos enlaces da rede $\mathbf{Y}$. A Figura 4 demonstra esse resultado após a realização de todas as tarefas desta etapa.

Quanto à criptografia homomórfica utilizada nesta etapa, é recomendado que se utilize um esquema de critptografia completamente homomórfica (*Fully Homomorphic*

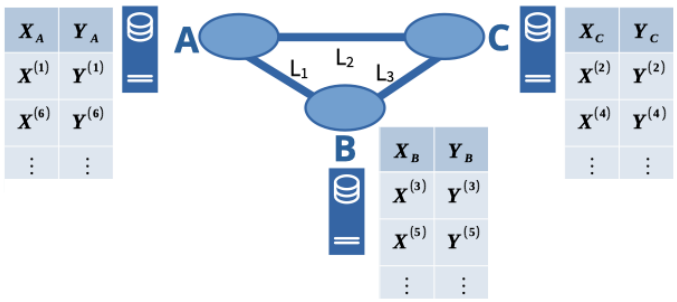

Figura 4. Matrizes de tráfego e medida de desempenho em cada nó.

Encryption – FHE), descrito em [Acar et al. 2018], para comportar múltiplas operações com os dados criptografados na próxima etapa.

Os autores de [Acar et al. 2018] mencionam o impacto no custo de processamento em esquemas de criptografia do tipo FHE. Uma característica deste método que auxilia na diminuição deste impacto é o fato da taxa de amostragem da coleta dos dados ser da ordem de minutos, assim como em [Grigoriou et al. 2018]. Por consequência, é possível dividir o processamento da criptografia FHE entre coletas adjacentes, que são relativamente espaçadas ao longo do tempo. Além disso, será flexibilizada, neste método, a criação de novas chaves de criptografia e a variedade da distribuição delas, para os dados de um mesmo nó. Esta flexibilização é feita para que o método seja capaz de realizar o aprendizado com os dados criptografados. Esta é outra característica que auxilia na redução de custo de processamento da criptografia FHE.

4.3. Etapa 3: Aprendizado federado

A principal tarefa desta etapa é a realização do treinamento de um modelo de predição de QoS de enlace de rede, através de uma técnica baseada em aprendizado federado. No aprendizado federado, a cada iteração, os nós da rede contribuem com um treinamento local usando os seus dados locais e o modelo global da iteração anterior. Um servidor global agrega todas as contribuições locais gerando um modelo global que será compartilhado com todos os nós, na iteração seguinte.

O conjunto de dados, do método descrito neste artigo, se trata de amostras de matrizes de tráfego com valores criptografados e que se encontram distribuídas entre os nós da rede. Essas matrizes de tráfego criptografadas devem apresentar valores constantes para as suas dimensões: C colunas e C linhas. Sendo assim, o vetor de entrada do modelo $\mathbf{X}$ deve ter dimensão $C \times C$, ou seja, $\mathbf{X} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_j)$, $j = C \times C$.

Na literatura, existem vários algoritmos baseados em aprendizado federado (FedSGD, FedAVG, etc.). O algoritmo de aprendizado federado será de escolha do projetista que aplica o método, assim como também as escolhas de taxa de aprendizado η e do tipo de função de perda p .

Após o aprendizado federado, o método converge para valores de parâmetros w muito próximos entre iterações adjacentes, de acordo com uma tolerância escolhida. A função de predição $f(\mathbf{X})$, então, pode ser aplicada para inferir a QoS nos enlaces da rede. Para que o método funcione, é necessário que os nós da rede repitam as suas mesmas regras de criptografia homomórfica para montar as matrizes de tráfego criptografadas $\mathbf{X}$.

5. Experimento e análises

Esta seção descreve as características do conjunto de dados que foi escolhido para a realização do experimento, detalhando as definições importantes para a implementação do método proposto. Por fim, há uma apresentação e análise dos resultados.

5.1. Conjunto de dados

O conjunto de dados que foi utilizado no experimento foi gerado pela mesma ferramenta que gerou os dados encontrados nos trabalhos [Amoroso et al. 2020, Uhlig et al. 2006] e encontra-se disponível em [2]. Esses dados se tratam de matrizes de trafego de uma rede acadêmica dos EUA, a rede Abilene. A topologia desta rede está ilustrada na Figura 5. Nela, é possível checar, ainda, os valores de custo associados aos enlaces pelo protocolo de roteamento interno usado na rede.

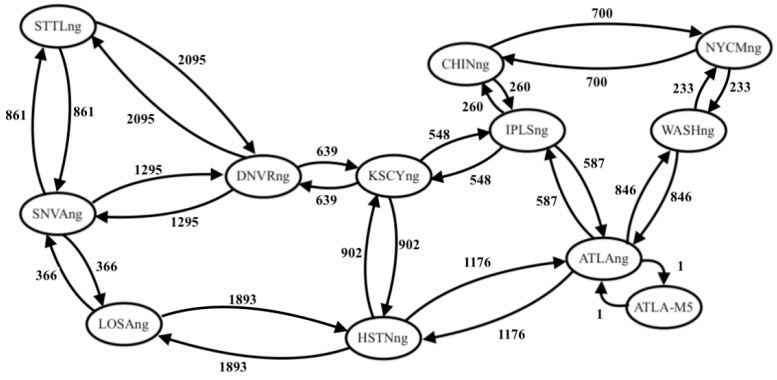

Figura 5. Topologia da rede Abilene e valores de atraso entre seus enlaces.

O conjunto inteiro de dados possui 48096 matrizes de tráfego. As matrizes de tráfego do conjunto de dados possuem 12 linhas e 12 colunas. Cada ponto de cada matriz representa a taxa de transferência de dados entre os nós da rede. Visto que os nós da Abilene representam *backbones* de rede, existem vários dispositivos conectados em cada nó e, portanto, há dados de tráfego que possuem o mesmo nó como origem e destino. Para determinado valor de taxa de transferência em cada ponto da matriz, a posição da linha l representa a origem e a posição da coluna c representa o destino, $0 \leq l < 12$ e $0 \leq c < 12$. Os valores de transferência são apresentados em kbits por segundo (kbps). A Figura 6 apresenta uma matriz de tráfego presente no conjunto de dados.

5.2. Implementação do método proposto

Antes de aplicar as matrizes do *dataset* diretamente no algoritmo de aprendizado federado, há um pré-processamento que deve ser realizado para reproduzir o cenário no qual o sistema se encontraria após a coleta de dados, conforme descrito na etapa 1. Este pré-processamento se trata da aplicação de uma regra de criptografia homomórfica para cada posição de linha l nas matrizes de tráfego. Isto simula a criptografia homomórfica escolhida por cada nó da rede.

Para simplificação do experimento, são simuladas criptografias homomórficas básicas, tais como a multiplicação por um valor constante, conforme as seguintes regras:

[2] `https://totem.run.montefiore.uliege.be/datatools.html`

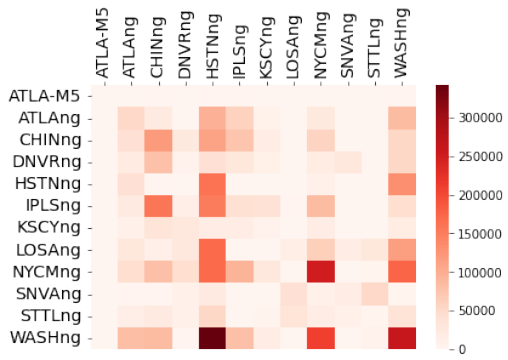

Figura 6. Matriz de tráfego do conjunto de dados.

Para valores em posições da matriz em que $l = 0$, logo o valor é multiplicado por 5; se $l = 1$, ele é multiplicado por 6; e assim, sucessivamente, até os valores das posições de $l = 11$, que são multiplicados por 16.

De acordo com a etapa 1 do método, é necessário aplicar o algoritmo de *Dijkstra* para encontrar os menores caminhos entre os nós da rede. Assim, a rotulação dos enlaces é feita, apenas, pela razão entre a soma das taxas das transferência que percorrem o enlace e sua capacidade de banda. Neste experimento, o aprendizado será realizado, apenas, com base no enlace de LOSAng para HSTNng. Logo, neste caso, o vetor de desempenho de enlaces Y é unitário, pelo fato de considerar apenas um desses enlaces, $Y = (y_{\text{LOSAng}\rightarrow\text{HSTNng}})$. Para cada amostra de matriz de tráfego, este único elemento $y_{\text{LOSAng}\rightarrow\text{HSTNng}}$ é calculado pela divisão entre a soma das taxas de transferências que passam pelo enlace LOSAng $\rightarrow$ HSTNng e a capacidade de banda passante deste enlace $bw_{\text{LOSAng}\rightarrow\text{HSTNng}} = 9920000$ kbps.

Por sua vez, o vetor de rótulos $\mathbf{Y}$ de QoS nos enlaces também é unitário. E o mapeamento do valor de desempenho y de enlace para o seu correspondente rótulo $\mathbf{y}$ de QoS se dá pela seguinte regra, definida neste experimento: se $y_{\text{LOSAng}\rightarrow\text{HSTNng}} \geq 100000$ kbps, então $\mathbf{y}_{\text{LOSAng}\rightarrow\text{HSTNng}} = 1$; caso contrário, $\mathbf{y}_{\text{LOSAng}\rightarrow\text{HSTNng}} = 0$.

O fim deste pré-processamento resulta num conjunto de dados com amostras que contêm o seguinte formato: um vetor de entrada $\mathbf{X}^{(t)} = (\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_{143}^{(t)})$ e vetor unitário de saída $\mathbf{Y}^{(t)} = (\mathbf{y}_{\text{LOSAng}\rightarrow\text{HSTNng}})$, tal que $0 \leq t < 48096$. Deste conjunto de dados, 38400 amostras são aleatoriamente separadas para serem usadas no treinamento. Enquanto, as amostras restantes são separadas para teste.

A partir deste ponto, serão realizados, paralelamente, dois tipos de treinamento: um deles utilizando o treinamento do método proposto neste artigo, sendo ele baseado em aprendizado federado, e o outro utilizando um treinamento centralizado baseado em gradiente descendente. Em ambos os treinamentos, a função de ativação q escolhida é uma função logística e ela define a função preditiva $f(\mathbf{X}^{(t)}) = q(w_0\mathbf{x}_0^{(t)} + \cdots + w_j\mathbf{x}_{143}^{(t)})$ do modelo. A função de perda $p(\mathbf{X}^{(t)})$, em ambos os treinamentos, é dada pelo erro quadrático em comparação com o rótulo de QoS correspondente $\mathbf{Y}^{(t)}$. A taxa de aprendizado η utilizada foi de $0,8$.

No aprendizado federado, as amostras separadas para treinamento são divididas entre 12 subconjuntos, representando os 12 nós da rede. O treinamento é realizado con-

forme a etapa 3 do método descrito neste artigo, usando o algoritmo FedAVG. Enquanto, no aprendizado centralizado, o treinamento é realizado usando gradiente descendente. Cada treinamento finaliza após a convergência dos valores de parâmetros w ao longo das iterações. Os valores finais de w são usados na função de predição $f(X^{(t)})$ para testar o modelo treinado, através do conjunto de dados separado para teste.

5.3. Métricas de avaliação

Para avaliar o método proposto neste artigo, primeiramente, ele deve ser testado através da aplicação da função preditiva $f(\mathbf{X}^{(t)})$ aprendida no treinamento. Para isso, os dados $X^{(t)}$, na parte do *dataset* separada para teste, são aplicados como entrada da $f(\mathbf{X}^{(t)})$. A resposta dessa função $f(\mathbf{X}^{(t)})$ é, então, comparada ao correspondente valor esperado $\mathbf{Y}^{(t)}$.

Durante a comparação entre a resposta da função preditiva $f(\mathbf{X}^{(t)})$ e o correspondente valor esperado $\mathbf{Y}^{(t)}$, ao longo de todo o conjunto de dados separados para teste, o sistema contabiliza as amostras de acordo com as seguintes classificações: uma amostra de teste representa um Verdadeiro Positivo (TP) se $f(\mathbf{X}^{(t)}) = 1$ e $\mathbf{Y}^{(t)} = 1$; representa um Verdadeiro Negativo (TN) se $f(\mathbf{X}^{(t)}) = 0$ e $\mathbf{Y}^{(t)} = 0$; um Falso Positivo (FP) se $f(\mathbf{X}^{(t)}) = 1$ e $\mathbf{Y}^{(t)} = 0$; e um Falso Negativo (FN) se $f(\mathbf{X}^{(t)}) = 0$ e $\mathbf{Y}^{(t)} = 1$.

De acordo com [Fawcett 2006], as métricas de avaliação de modelos, chamadas de acurácia, precisão, *recall* e *f1 score*, podem ser calculadas a partir da quantidade de amostras TP, TN, FP e FN em cada classificação. Logo, o cálculo das métricas mencionadas é dado por: acurácia $= \frac{TP+TN}{TP+FP+TN+FN}$; precisão $= \frac{TP}{TP+FP}$; *recall* $= \frac{TP}{TP+FN}$; *f1 score* $= \frac{2 \times \text{precisão} \times \textit{recall}}{\text{precisão} + \textit{recall}}$.

5.4. Resultados e análises

Através do processo de teste do método proposto, são computadas as métricas acurácia, precisão, *recall* e *f1 score*. Para fins de comparação, foi também treinado e testado um modelo com base em aprendizado centralizado. A Tabela 1 apresenta uma comparação entre as métricas de avaliação do aprendizado federado e as do aprendizado centralizado.

Tabela 1. Comparação entre as métricas de avaliação dos 2 tipos de aprendizado.

	Aprendizado federado	Aprendizado centralizado
acurácia	**0,82931106**	0,80858086
precisão	**0,80995195**	0.77175157
recall	0,92412451	**0.95383799**
f1 score	**0,86327964**	0.85318779

Além da comparação descrita acima, o experimento utilizando o aprendizado federado foi capaz de treinar o modelo com menos rodadas do que o mesmo experimento utilizando aprendizado centralizado. A Figura 7 apresenta graficamente a evolução do valor de precisão do modelo, ao longo das rodadas de treinamento, utilizando aprendizado federado e aprendizado centralizado. Uma convergência do modelo através de menos rodadas de treinamento, ao utilizar o aprendizado federado, também é demonstrada de forma semelhante em [Amoroso et al. 2020]. Ao invés da apresentação da evolução da precisão ao longo das rodadas, os autores demonstram que, no aprendizado federado, há um decaimento do erro médio absoluto e sua estabilização em menos rodadas do que no aprendizado centralizado.

Por fim, é importante dizer que o método proposto no presente artigo, além de alcançar resultados positivos em comparação ao aprendizado centralizado, permite também uma maior segurança aos dados dos usuários da rede graças o uso de criptografia homomórfica.

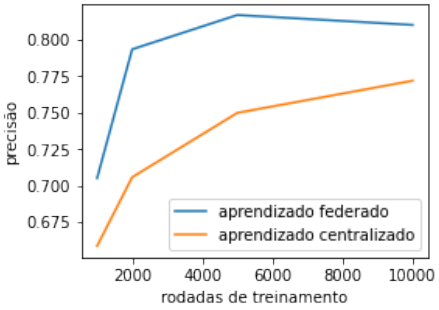

Figura 7. Comparação da evolução da precisão entre os diferentes aprendizados.

6. Conclusão

Este trabalho propôs um método para inferir a QoS na rede a partir de matrizes de tráfego com medições criptografadas. Os resultados mostraram que o modelo proposto foi capaz de realizar tal inferência com acurácia de aproximadamente 83% e com precisão de aproximadamente 81%. Estas métricas de avaliação do método proposto superaram o desempenho de avaliação de uma abordagem centralizada. Além disto, o método proposto neste artigo traz a vantagem de manter a privacidade dos dados, através do uso de criptografia homomórfica nas medidas de tráfego dos nós da rede.

Em trabalhos futuros, será considerada a utilização de outros *datasets*, além de uma investigação a respeito do impacto de diferentes valores de parâmetros para melhorar as métricas de desempenho. Também com relação à criptografia homomórfica, serão realizados estudos com técnicas mais sofisticadas e o impacto destas técnicas na solução.

Referências

[Acar et al. 2018] Acar, A., Aksu, H., Uluagac, A. S., and Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys*, 51(4):1–35.

[Amoroso et al. 2020] Amoroso, R., Esposito, F., and Merani, M. L. (2020). Estimation of traffic matrices via super-resolution and federated learning. In *16th ACM CoNEXT*, pages 560–561.

[Aono et al. 2017] Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. (2017). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345.

[Bishop and Nasrabadi 2006] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

[Clausen et al. 2022] Clausen, H., Manocha, A., and Gibson, M. (2022). Detecting proxies relaying streaming internet traffic. In *IEEE INFOCOM WKSHPS*, pages 1–6.

[Fawcett 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

[Goodfellow et al. 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Grigoriou et al. 2018] Grigoriou, E., Saoulidis, T., Atzori, L., Pilloni, V., and Chatzimisios, P. (2018). An agent-based qoe monitoring strategy for lte networks. In *IEEE ICC*, pages 1–6.

[Hartmann 2018] Hartmann, F. (2018). Federated learning. Master's thesis, Institut fur Informatik der Freien Universität Berlin, Available on: https://florian.github.io/federated-learning/.

[Lo et al. 2021] Lo, S. K., Lu, Q., Wang, C., Paik, H.-Y., and Zhu, L. (2021). A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Computing Surveys*, 54(5):1–39.

[McMahan et al. 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In *20th AISTATS*, pages 1273–1282.

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

[Mesquita and Assis 2019] Mesquita, L. A. J. and Assis, K. D. R. (2019). Traffic matrix prediction for optical networks. In *IEEE IMOC*, pages 1–3.

[Monteiro et al. 2019] Monteiro, N., Fontinele, A., Santos, I., Costa, A., Campelo, D., and Soares, A. (2019). Novo algoritmo para provisão de banda de guarda adaptativa em redes Ópticas elásticas. In *XXXVII SBRC*, pages 307–320.

[Navarro-Ortiz et al. 2020] Navarro-Ortiz, J., Romero-Diaz, P., Sendra, S., Ameigeiras, P., Ramos-Munoz, J. J., and Lopez-Soler, J. M. (2020). A survey on 5g usage scenarios and traffic models. *IEEE Communications Surveys & Tutorials*, 22(2):905–929.

[She et al. 2022] She, H., Zhu, X., Guo, Y., Cao, H., Garg, S., and Kaddoum, G. (2022). Bclb: Blockchain-based controller load balance for safe and reliable resource optimization. In *IEEE Globecom Workshops*, pages 668–673.

[Thonglek et al. 2020] Thonglek, K., Takahashi, K., Ichikawa, K., Iida, H., and Nakasan, C. (2020). Federated learning of neural network models with heterogeneous structures. In *19th IEEE ICMLA*, pages 735–740.

[Tu et al. 2015] Tu, R., Wang, X., Zhao, J., Yang, Y., Shi, L., and Wolf, T. (2015). Design of a load-balancing middlebox based on sdn for data centers. In *IEEE INFOCOM WKSHPS*, pages 480–485.

[Uhlig et al. 2006] Uhlig, S., Quoitin, B., Lepropre, J., and Balon, S. (2006). Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86.