Projeto para a Disciplina de Métodos Numéricos
Curso: Ciências da Computação
NumFlow: Análise e implementação de métodos para estipulação de fluxo de dados em redes de Computadores

Alunos: Pedro Daniel de Oliveira Hoeller e Gianluca Matos Klaumann
Professor: Luis Fernando Nazari
pdanieldeoliveirahoeller@gmail.com, gianlucamk04@gmail.com, luis.nazari@ifc.edu.br

8 de setembro de 2025

Resumo

Este artigo apresenta uma abordagem para a análise do fluxo de dados em redes de computadores, utilizando métodos numéricos para a resolução de sistemas lineares. A pesquisa propõe um algoritmo flexível que, a partir da adaptação de uma única matriz A, permite a modelagem de três contextos distintos: Mbps (Megabits por segundo), RPS (requisições por segundo) e latência. Em cada modelo, os coeficientes da matriz, o vetor de demandas e as variáveis de saída são interpretados de forma a refletir aspectos específicos do tráfego de rede, possibilitando a identificação de problemas como sobrecarga, quedas de servidores e gargalos na comunicação. Para a resolução dos sistemas lineares, foram empregados métodos numéricos, tais como o escalonamento de Gauss com pivoteamento e refinamento, além dos métodos iterativos de Jacobi e Gauss-Seidel, que foram implementados e comparados quanto à sua eficácia e precisão. Os resultados obtidos evidenciam que, embora o método direto apresente alta precisão e convergência em diferentes topologias de rede, os métodos iterativos podem enfrentar dificuldades de convergência em cenários onde as matrizes não são diagonalmente dominantes. Assim, a análise comparativa entre as três abordagens permite identificar a metodologia mais adequada para cada contexto, contribuindo para o aprimoramento das técnicas de monitoramento, gestão e balanceamento de carga em ambientes computacionais.

1 Introdução

De acordo com Bueno e Weingartner, com o aumento contínuo da demanda pela Internet, o gerenciamento de redes passou a ser uma parte essencial do cotidiano no setor de tecnologia da informação. Para uma administração eficaz, é fundamental garantir um fluxo adequado de informações, a utilização correta dos recursos disponíveis, a ausência de sobrecarga e a manutenção da confiabilidade e segurança na transmissão de dados. Assim, é notória a importância de adotar métodos que possibilitem uma análise detalhada e precisa do tráfego de dados em redes de computadores.

Nesse contexto, a presente pesquisa tem como objetivo analisar o fluxo de informações por meio de métodos numéricos, empregando algoritmos baseados em escalonamento de Gauss, Jacobi e Gauss-Seidel para a resolução de sistemas lineares. A abordagem proposta permite a modelagem de diferentes aspectos do tráfego, possibilitando a identificação de problemas como queda de servidores, sobrecarga, distribuição inadequada da carga e congestionamento, além de oferecer subsídios para a avaliação do estado operacional dos nós da rede.

O algoritmo desenvolvido destaca-se por sua flexibilidade, possibilitando a adaptação de uma matriz A para a modelagem de três contextos distintos: Mbps (Megabits por segundo), RPS (requisições por segundo) e latência. No modelo Mbps, o tráfego é mensurado em Megabits por segundo, em que os coeficientes A_{ij} representam a capacidade dos links ou o peso da comunicação entre servidores, os termos b_i correspondem à quantidade de dados recebida por cada servidor, e os x_i indicam o fluxo de saída. No cenário de RPS, os elementos A_{ij} são interpretados como o peso da carga computacional entre os servidores, b_i indicam o número de requisições recebidas, e x_i representam as requisições

processadas por segundo. Por fim, o modelo de latência utiliza A_{ij} para indicar o tempo médio de comunicação entre servidores, b_i para a latência total esperada por cliente, e x_i para a latência efetivamente percebida em cada nó. As três opções serão implementadas e comparadas entre si, de forma a identificar a abordagem mais adequada para cada contexto analisado.

A escolha dos métodos numéricos para a resolução dos sistemas lineares oriundos dessa modelagem justifica-se pela eficiência na obtenção de soluções que revelam não apenas a dinâmica do fluxo de dados, mas também informações críticas quanto ao desempenho e à integridade dos servidores. Dessa forma, o estudo contribui para o aprimoramento das técnicas de monitoramento e gestão de redes, fortalecendo a fundamentaçãFormulação Matemáticao para a tomada de decisões estratégicas em ambientes computacionais.

2 Metodologia

2.1 Formulação Matemática

A modelagem do tráfego de dados em redes de computadores pode ser formalizada por meio de sistemas lineares da seguinte forma:

$$Ax = b$$

Onde a matriz A representa os parâmetros e as características dos links de comunicação, o vetor x denota as variáveis associadas ao fluxo de saída dos servidores e o vetor b expressa a demanda ou a carga de cada nó da rede.

Para adaptar a modelagem a diferentes contextos, definem-se as seguintes interpretações para os elementos da equação:

Modelo Mbps

Neste cenário, o tráfego é mensurado em Megabits por segundo. Assim, temos:

- A_{ij}: representa a capacidade dos links ou o peso da comunicação entre os servidores i e j;
- b_i: corresponde à quantidade de dados recebida pelo servidor i;
- x_i: indica o fluxo de dados que sai do servidor i.

O sistema de equações é expresso por:

$$\sum_{j=1}^{n} A_{ij}x_j = b_i, \quad \text{para } i = 1, 2, \ldots, n.$$

Modelo RPS

No modelo baseado em requisições por segundo, os elementos são interpretados da seguinte forma:

- A_{ij}: representa o peso da carga computacional entre os servidores i e j;
- b_i: indica o número de requisições por segundo recebidas pelo servidor i;
- x_i: representa o número de requisições processadas por segundo pelo servidor i.

A modelagem mantém a estrutura do sistema linear:

$$\sum_{j=1}^{n} A_{ij}x_j = b_i, \quad \text{para } i = 1, 2, \ldots, n.$$

Modelo de Latência

Para a análise da latência na rede, os parâmetros assumem os seguintes significados:

- A_{ij}: denota o tempo médio de comunicação entre os servidores i e j;
- b_i: representa a latência total esperada por cliente para o servidor i;
- x_i: indica a latência efetivamente percebida em cada nó i.

Novamente, o sistema é formulado como

$$\sum_{j=1}^{n} A_{ij} x_j = b_i, \quad \text{para } i = 1, 2, \ldots, n.$$

Em todas as três abordagens, a estrutura do sistema linear permanece a mesma, permitindo que o algoritmo desenvolvido seja adaptado facilmente ao alterar a interpretação dos coeficientes e das variáveis de acordo com o contexto analisado. A resolução desses sistemas será realizada utilizando métodos numéricos, como o escalonamento de Gauss e os métodos iterativos de Jacobi e Gauss-Seidel, que garantem a obtenção de soluções eficientes e convergentes sob condições apropriadas.

Essa formulação matemática serve de base para a análise comparativa dos três modelos. Ao aplicar os métodos numéricos, é possível identificar a abordagem mais adequada para cada cenário, levando em conta aspectos críticos como a detecção de sobrecarga, a distribuição inadequada da carga, quedas de servidores e possíveis indicações de congestionamento, contribuindo assim para o aprimoramento das técnicas de monitoramento e gestão de redes.

2.2 Implementação Numérica

Nesta seção, serão discutidos três métodos clássicos para a resolução de sistemas lineares: o Método de Gauss com Pivoteamento e Refinamento, o Método de Jacobi e o Método de Gauss-Seidel. Cada um desses métodos apresenta suas vantagens e limitações, dependendo das características do sistema em questão, como a convergência e a precisão exigida.

2.2.1 Método de Gauss com pivoteamento e refinamento

Este método é uma variação do tradicional método de eliminação de Gauss, no qual é aplicado o pivoteamento para evitar divisões por valores próximos de zero, garantindo maior estabilidade numérica. Além disso, a etapa de refinamento é utilizada para melhorar a precisão da solução obtida, especialmente em sistemas mais complexos. Segue abaixo a implementação realizada em Python:

```python
def eliminacao_gauss(A, b, max_iter):
    n = len(A)
    # Etapa de eliminação de Gauss
    for i in range(n):
        # Encontra o índice da linha com o maior valor absoluto no elemento A[i]
        max_index = max(range(i, n), key=lambda k: abs(A[k][i]))
        if A[max_index][i] == 0:
            raise ValueError(f"A matriz é singular, não é possível resolver o sistema para a linha {i}.")
        # Troca a linha atual (i) com a linha que tem o maior valor absoluto na coluna
        if max_index != i:
            A[i], A[max_index] = A[max_index], A[i]
            b[i], b[max_index] = b[max_index], b[i]
        # Elimina os elementos abaixo do pivô
        for k in range(i + 1, n):
            fator = A[k][i] / A[i][i]  # Calcula o fator de eliminação
            for j in range(i, n):
                A[k][j] -= fator * A[i][j]
            b[k] -= fator * b[i]
    # Substituição retroativa
    x = [0] * n
    for i in range(n - 1, -1, -1):
        soma = sum(A[i][j] * x[j] for j in range(i + 1, n))
        x[i] = (b[i] - soma) / A[i][i]
    # Etapa de refinamento
```

```
    for _ in range(max_iter):
        # Calcula o vetor resíduo r, que é a diferença entre o lado direito e a multiplicação de A com x
        r = [b[i] - sum(A[i][j] * x[j] for j in range(n)) for i in range(n)]
        delta_x = [0] * n
        # Calculo da correção para cada x[i]
        for i in range(n - 1, -1, -1):
            soma = sum(A[i][j] * delta_x[j] for j in range(i + 1, n))
            delta_x[i] = (r[i] - soma) / A[i][i]
        # Atualiza a solução x
        x = [x[i] + delta_x[i] for i in range(n)]
    return x
```

Listing 1: Implementação do método de Eliminação de Gauss com Pivoteamento e Refinamento

2.2.2 Método de Jacobi

O Método de Jacobi é uma técnica iterativa simples, que busca uma solução aproximada através de iterações sucessivas. Cada variável do sistema é atualizada em cada iteração com base nas soluções obtidas na iteração anterior, e o processo continua até que a solução converja dentro de uma margem de erro desejada. Segue abaixo a implementação:

```
def metodo_jacobi(A, b, max_iter, tol=1e-6):
    n = len(A)
    x = [0] * n
    x_novo = x[:]
    # Laço de iteração para até o número máximo de iterações
    for _ in range(max_iter):
        # Laço que percorre cada equação do sistema
        for i in range(n):
            # Calcula a soma dos produtos dos coeficientes por seus valores anteriores de x, excluindo A[i][i]
            soma = sum(A[i][j] * x[j] for j in range(n) if j != i)
            # Atualiza x_novo com o novo valor calculado para a variável x[i]
            x_novo[i] = (b[i] - soma) / A[i][i]
        # Verifica se a diferença máxima entre os valores antigos e novos é menor que a tolerância
        if max(abs(x_novo[i] - x[i]) for i in range(n)) < tol:
            # Se a diferença for menor que a tolerância, retorna a solução encontrada
            return x_novo
        x = x_novo[:]
    return x_novo
```

Listing 2: Implementação do método de Jacobi

2.2.3 Método de Gauss-Siedel

Semelhante ao Método de Jacobi, o Método de Gauss-Seidel também é iterativo, mas, em vez de usar valores antigos de variáveis em cada iteração, ele utiliza os valores mais recentes para atualizar as variáveis do sistema. Esse comportamento geralmente proporciona uma convergência mais rápida em relação ao método de Jacobi, mas pode exigir cuidados adicionais quanto à escolha do ponto de partida e da taxa de convergência. Segue portanto o código em Python da implementação:

```
def metodo_gauss_seidel(A, b, max_iter, tol=1e-6):
    n = len(A)
    x = [0] * n
    for _ in range(max_iter):  # Laço para as iterações
        x_old = x[:]
        for i in range(n):  # Itera sobre cada variável
            soma = sum(A[i][j] * x[j] for j in range(n) if j != i)
            x[i] = (b[i] - soma) / A[i][i]
        if max(abs(x[i] - x_old[i]) for i in range(n)) < tol:  # Verifica se a mudança é menor que a tolerância
            return x
    return x
```

Listing 3: Implementação do método de Gauss-Siedel

2.3 Métricas de Erro

Ao avaliar um modelo numérico, seja ele baseado em métodos diretos, como a eliminação de Gauss, ou métodos iterativos, como Jacobi e Gauss-Seidel, é essencial medir o quão próxima a solução calculada $\mathbf{x}_{\text{calc}}$ está da solução exata, ou verificar a precisão com que o sistema $\mathbf{Ax} = \mathbf{b}$ é satisfeito. No entanto, para o modelo adotado, não há uma solução exata disponível para comparação e validação do método.

Portanto, a avaliação do erro deve ser feita por meio das métricas de erro absoluto e relativo em relação à solução obtida. Para isso fora implementado em Python 2 funções específicas para calcular os erros, segue abaixo a implementação das funções e uma breve explicação.

```
def Ax(A, x):
    # Para cada linha da matriz A, calcula a soma ponderada dos elementos por x
    return [sum(A[i][j] * x[j] for j in range(len(x))) for i in range(len(A))]
def erroAbsoluto(A, b, x):
    # Calcula a diferença absoluta entre cada elemento de Ax(A, x) e b
    return [abs(Ax(A, x)[i] - b[i]) for i in range(len(b))]
def erroRelativo(erroAbsoluto, b):
    # Para cada elemento em b, calcula o erro relativo, tratando divisão por zero
    return [erroAbsoluto[i] / abs(b[i]) if b[i] != 0 else 0 for i in range(len(b))]
```

Listing 4: Implementação do método de Eliminação de Gauss com Pivoteamento e Refinamento

3 Resultados e discussão

Neste módulo, serão apresentadas as análises de desempenho e precisão no modo de Mbps para diferentes topologias de redes: Árvore, Barramento, Estrela e Ponto a Ponto. A análise tem como objetivo avaliar como cada topologia afeta o desempenho do fluxo de dados entre os servidores, considerando a eficiência geral do sistema. Para tal, foram implementados modelos matemáticos que representam as interações de comunicação entre os servidores de forma computacional, possibilitando uma comparação entre as topologias. A base para a análise individual de cada topologia de rede foi construída por meio da implementação de uma matriz A de coeficientes, que modela as interações entre os servidores conectados na rede. Cada elemento dessa matriz reflete a intensidade ou capacidade da comunicação entre dois servidores específicos. Os coeficientes foram cuidadosamente ajustados de acordo com a estrutura de cada topologia, permitindo uma análise detalhada das dinâmicas de fluxo de dados

3.1 Valores de demandas de dados, Matriz B

Para podermos avaliar o desempenho entre as topologias de rede optamos por utilizar a mesma matriz de demanda de dados para compararmos as melhores distribuições de processamento de dados, portanto definimos a matriz como: $b = [500, 300, 700, 350, 400]$

3.2 Árvore

Na topologia de árvore, temos um nó raiz com ramificações que se estendem até os nós folhas. Ela é hierárquica, com comunicação entre os nós sempre passando por um nó superior. É comumente utilizada em cenários em que é necessário um controle centralizado e quando se busca maior eficiência em termos de largura de banda.

3.2.1 Grafo da Matriz de A

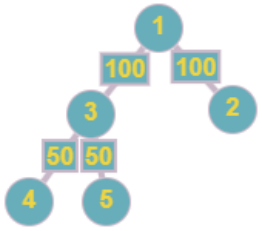

Figura 1: Grafo da topologia de rede em árvore

3.2.2 Solução do sistema

Para as entradas anteriores de dados a solução encontrada do sistema foi:

$x = [0.76905311778291, 0.5577367205542725, 1.3660508083140877, 0.704243648960739, 0.8292436489607391]$

3.2.3 Erros do sistema

Segue abaixo o valor de métricas de erro do sistema para cada método:

Método/Erro	Erro Absoluto	Erro Relativo
Gauss com pivoteamento e refinamento	$[0.0, 0.0, 0.0, 0.0, 0.0]$	$[0.0, 0.0, 0.0, 0.0, 0.0]$
Jacobi	$[203.81, 39.28, 75.97, 5.55, 0.0]$	$[0.41, 0.22, 0.13, 0.02, 0.0]$
Gauss-Seidel	$[203.81, 39.28, 75.97, 5.55, 0.0]$	$[0.41, 0.22, 0.13, 0.02, 0.0]$

Tabela 1: Comparação entre métodos numéricos e seus respectivos erros

3.3 Barramento

A topologia de barramento conecta todos os nós a um único cabo, conhecido como barramento. Os dados são enviados ao longo deste barramento e todos os nós recebem a mesma mensagem, mas apenas o nó de destino processa a mensagem.

3.3.1 Grafo da Matriz de A

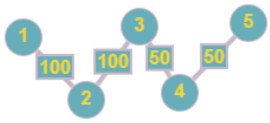

Figura 2: Grafo da topologia de rede em barramento

3.3.2 Solução do sistema

Para as entradas anteriores de dados a solução encontrada do sistema foi:

$x = [1.2450980392156863, 0.01960784313725488, 1.6764705882352942, 0.5490196078431372, 0.9313725490196079]$

3.3.3 Erros do sistema

Segue abaixo o valor de métricas de erro do sistema para cada método:

Método/Erro	Erro Absoluto	Erro Relativo
Gauss com pivoteamento e refinamento	$[0.0, 0.0, 0.0, 0.0, 0.0]$	$[0.0, 0.0, 0.0, 0.0, 0.0]$
Jacobi	$[46.67, 175.0, 33.37, 46.57, 0.0]]$	$[0.09, 1.0, 0.05, 0.18, 0.0]$
Gauss-Seidel	$[46.67, 175.0, 33.37, 46.57, 0.0]$	$[0.09, 1.0, 0.05, 0.18, 0.0]$

Tabela 2: Comparação entre métodos numéricos e seus respectivos erros

3.4 Estrela

Em uma topologia estrela, todos os nós são conectados a um nó central (geralmente um switch ou roteador), que é responsável por coordenar a comunicação entre os dispositivos. A falha no nó central pode causar a interrupção de toda a rede.

3.4.1 Grafo da Matriz de A

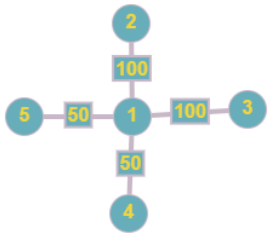

Figura 3: Grafo da topologia de rede em estrela

3.4.2 Solução do sistema

Para as entradas anteriores de dados a solução encontrada do sistema foi:

$x = [0.4629629629629629, 0.6342592592592593, 1.6342592592592593, 0.8171296296296297, 0.9421296296296298]$

3.4.3 Erros do sistema

Segue abaixo o valor de métricas de erro do sistema para cada método:

Método/Erro	Erro Absoluto	Erro Relativo
Gauss com pivoteamento e refinamento	$[0.0, 0.0, 0.0, 0.0, 0.0]$	$[0.0, 0.0, 0.0, 0.0, 0.0]$
Jacobi	$[290.92, 61.06, 23.23, 6.73, 0.0]$	$[0.58, 0.35, 0.04, 0.02, 0.0]$
Gauss-Seidel	$[290.92, 61.06, 23.23, 6.73, 0.0]$	$[0.58, 0.35, 0.04, 0.02, 0.0]$

Tabela 3: Comparação entre métodos numéricos e seus respectivos erros

3.5 Ponto a Ponto

A topologia ponto a ponto conecta dois dispositivos diretamente sem intermediários. É uma das topologias mais simples e eficazes para conexões diretas, como em links dedicados.

3.5.1 Grafo da Matriz de A

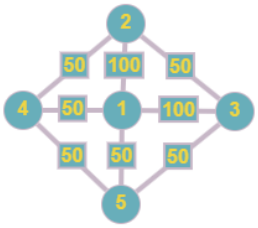

Figura 4: Grafo da topologia de rede em ponto a ponto

3.5.2 Solução do sistema

Para as entradas anteriores de dados a solução encontrada do sistema foi:

$x = [0.6847133757961784, 0.12544663663197142, 1.4669100512661177, 0.6931800528196363, 0.6443995650147585]$

3.5.3 Erros do sistema

Segue abaixo o valor de métricas de erro do sistema para cada método:

Método/Erro	**Erro Absoluto**	**Erro Relativo**
Gauss com pivoteamento e refinamento	$[0.0, 0.0, 0.0, 0.0, 0.0]$	$[0.0, 0.0, 0.0, 0.0, 0.0]$
Jacobi	$[267.45, 133.37, 10.34, 30.42, 0.0]$	$[0.53, 0.76, 0.01, 0.10, 0.0]$
Gauss-Seidel	$[267.45, 133.37, 10.34, 30.42, 0.0]$	$[0.53, 0.76, 0.01, 0.1, 0.0]$

Tabela 4: Comparação entre métodos numéricos e seus respectivos erros

3.6 Interpretação

A seguir, abordaremos a interpretação dos valores obtidos pelo modelo matemático, analisando suas implicações e relevância para o problema em questão.

3.6.1 Valores de x

Seja o sistema representado por:

$$A\mathbf{x} = \mathbf{b}$$

onde:

- A é a matriz de coeficientes, representa conectividade entre servidores, capacidade de processamento ou atrasos na rede.
- $\mathbf{x}$ é o vetor solução, que indica os valores do fluxo após resolver o sistema.
- $\mathbf{b}$ é o vetor de demanda ou carga de entrada, representando a quantidade de tráfego ou requisições recebidas.

A interpretação dos valores de $\mathbf{x}$ pode ser feita da seguinte forma:

Valor de x_i	Significado
$x_i > 0$	Indica que o servidor i está processando ou enviando dados.
$x_i \approx 0$	O servidor i tem pouca ou nenhuma carga de trabalho.
$x_i < 0$	Pode indicar congestionamento, fluxo reverso ou erro na modelagem.

3.6.2 Conjunto de valores

Se os valores de **x** representam a quantidade de requisições processadas por servidores, podemos interpretar o conjunto de valores de **x** como Balanceamento de Carga, assim sendo:

Cenário	Interpretação
Todos os x_i são próximos entre os servidores.	A carga está bem distribuída
Alguns x_i são muito maiores que outros servidores e possíveis gargalos.	Indica sobrecarga em alguns
$x_i = 0$ para algum i pode estar ocioso ou offline.	Esse servidor não está processando requisições,

3.7 Interpretação dos Resultados

A análise dos resultados obtidos para as diferentes topologias de rede revela informações cruciais sobre a eficácia dos métodos numéricos aplicados. Os métodos diretos, como o de Gauss com pivoteamento e refinamento, apresentaram erros nulos em todas as topologias, evidenciando sua alta confiabilidade em sistemas lineares bem condicionados, mesmo em matrizes densas ou mal condicionadas.

Por outro lado, os métodos iterativos, como Jacobi e Gauss-Seidel, falharam em convergir, apresentando erros elevados, como no caso do erro relativo de 100% no Barramento. Isso sugere que as matrizes associadas às topologias analisadas não são diagonalmente dominantes ou possuem um raio espectral desfavorável, o que dificulta a convergência desses métodos. A falta de convergência e a equivalência dos erros entre Jacobi e Gauss-Seidel indicam limitações nos critérios de parada.

Em relação à estrutura das matrizes, as topologias Árvore e Estrela, com fortes hierarquias, resultaram em matrizes com acoplamentos fortes entre variáveis, gerando erros significativos nos primeiros nós, especialmente devido à propagação de inexatidões a partir do nó raiz. No Barramento, a conexão única gerou sistemas com maior densidade de interações, refletindo-se em erros críticos, como no segundo nó, onde o erro relativo foi de 100%. A topologia Ponto a Ponto, embora simples, também não garantiu a convergência dos métodos iterativos.

Em termos práticos, os métodos diretos são recomendados para redes complexas, como as com forte hierarquia, evitando erros graves em aplicações como balanceamento de carga. Já os métodos iterativos, como Jacobi e Gauss-Seidel, devem ser aplicados com cautela, sendo essencial garantir características como diagonal dominância para evitar soluções imprecisas.

Uma observação interessante é que, nas topologias analisadas, a última variável apresentou erro absoluto zero nos métodos iterativos, sugerindo que a equação associada a ela é trivial e pode ser resolvida exatamente, mesmo sem a convergência dos métodos para as demais variáveis.

4 Conclusão

A análise dos métodos numéricos aplicados demonstrou que a interpretação dos resultados é fundamental para identificar gargalos e problemas na distribuição de carga dos servidores. Os valores obtidos no vetor **x** possibilitam verificar se a carga está bem distribuída ou se há sobrecarga em determinados nós, apontando possíveis falhas na rede. As métricas de erro, tanto absolutas quanto relativas, ajudam a validar a eficácia dos métodos, evidenciando que o método de Gauss com pivoteamento e refinamento apresenta alta precisão, enquanto os métodos iterativos (Jacobi e Gauss-Seidel) podem enfrentar dificuldades de convergência em certas topologias.

Em resumo, essa abordagem permite diagnosticar rapidamente pontos críticos na rede, como sobrecargas ou nós inativos, e fornece subsídios para a tomada de decisões estratégicas na gestão do fluxo de dados e balanceamento de carga, contribuindo para a melhoria da performance e confiabilidade dos sistemas computacionais.

5 Repositório

O código está disponível em: https://github.com/Pedro-IFC/NumFlow,
NumFlow: Análise e implementação de métodos para estipulação de fluxo de dados em redes de Computadores

Referências

[1] SILVA, Michael Angelo Alves da. *Análise e classificação de problemas de transmissão de dados em redes de computadores*. 2022. 79 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) Universidade Federal do Rio Grande do Norte, Caicó, 2022. Disponível em: https://repositorio.ufrn.br/handle/123456789/49429. Acesso em: 4 fev. 2025.

[2] SCHULTZ, Kelly Cristina. *Implementação e análise de uma estrutura de rede, contemplando gerenciamento, qualidade de serviços e segurança*. 2013. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) Universidade Tecnológica Federal do Paraná, Curitiba, 2013. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/9283/2/CT_COBSI_2013_1_04.pdf. Acesso em: 4 fev. 2025.

[3] IBM. *Análise de fluxo para redes modernas*. Disponível em: https://www.ibm.com/br-pt/products/sevone/flow-analysis. Acesso em: 4 fev. 2025.

[4] LOCAWEB. *Fluxo de dados: como usar suas principais ferramentas*. Disponível em: https://www.locaweb.com.br/blog/temas/codigo-aberto/fluxo-de-dados-principais-ferramentas. Acesso em: 4 fev. 2025.

[5] BUENO, Lucas Eduardo Neuhaus; WEINGARTNER, Thiago da Silva. *Monitoramento de dispositivos em rede utilizando Zabbix e Grafana*. 2020. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) Instituto Federal Farroupilha, Panambi, 2020. Disponível em: https://arandu.iffarroupilha.edu.br/bitstream/itemid/254/1/TCC%20Versao%20Final%20LUCAS%20EDUARDO.pdf. Acesso em: 4 fev. 2025.