# Da Sinapse à Borda com MI-X: Escoltando o Usuário no Edge-Cloud-Continuum

**Rodrigo A. Bezerra[1], Luiz Bittencourt[2], Leandro Villas[2], Maycon L. M. Peixoto [1]**

[1] Instituto de Computação – Universidade Federal da Bahia (UFBA)

{rodrigo.bezerra,maycon.leone}@ufba.br

[2]Instituto de Computação – Universidade Estadual de Campinas - UNICAMP
{bit, lvillas}@unicamp.br

***Abstract.*** *Current Fog Computing strategies struggle to meet the dynamic computational demands and strict low-latency requirements of real-time Brain-Computer Interfaces (BCIs), hindering efficient resource allocation and consistent performance under user mobility and changing network conditions. To address this challenge, this work presents a strategy for allocating latency-sensitive modular BCIs applications in a Hierarchical Fog Computing architecture, instantiated in this study through an EEG (electroencephalogram) Game scenario. The approach minimizes latency and response times while optimizing computational resource utilization. We develop a dynamic allocation algorithm, evaluate it under different mobility models, and assess its performance across multiple fog computing configurations. Results demonstrate reduced latency and more efficient resource usage, confirming the effectiveness of the proposed approach in fog–edge–cloud environments.*

***Resumo.*** *As estratégias atuais de Computação em Névoa enfrentam dificuldades para atender às demandas computacionais dinâmicas e aos rigorosos requisitos de baixa latência de Interfaces Cérebro–Computador (BCI), prejudicando a alocação eficiente de recursos e a consistência do desempenho sob mobilidade dos usuários e condições de rede em constante mudança. Para enfrentar esse desafio, este trabalho apresenta uma estratégia para a alocação de aplicações BCI modulares sensíveis à latência em uma arquitetura hierárquica de Computação em Névoa, instanciada neste estudo por meio de um cenário de jogo baseado em EEG (eletroencefalograma). A abordagem minimiza a latência e os tempos de resposta ao mesmo tempo em que otimiza a utilização dos recursos computacionais. Desenvolvemos um algoritmo de alocação dinâmica, avaliando-o sob diferentes modelos de mobilidade e analisando seu desempenho em múltiplas configurações de computação em névoa. Os resultados demonstram redução da latência e uso mais eficiente de recursos, confirmando a eficácia da abordagem proposta em ambientes névoa–borda–nuvem.*

## 1. Introdução

A integração de dispositivos IoT com Interfaces Cérebro–Computador (BCI) impõe requisitos rigorosos de processamento em tempo real [Li et al. 2015]. Aplicações interativas, como *EEG Games*, demandam sincronia precisa entre a intenção motora e o feedback visual, tornando a latência um requisito funcional essencial para a Qualidade de Experiência

(QoE) [Xu et al. 2023, Bittencourt et al. 2017]. A Computação em Névoa mitiga essas limitações ao aproximar o processamento da borda. Contudo, sessões dinâmicas e a mobilidade contínua dos usuários exigem mecanismos de orquestração capazes de adaptar-se a variações topológicas sob latência ultra-baixa [Bittencourt and Oliveira 2017].

Embora arquiteturas hierárquicas (Nuvem, Névoa e Cloudlets) melhorem a escalabilidade [Peixoto et al. 2021, Oliveira et al. 2024], muitas abordagens tratam a mobilidade de forma limitada, carecendo de mecanismos robustos para realocação dinâmica de módulos. Como aplicações BCI possuem módulos interdependentes com requisitos heterogêneos, a migração indiscriminada pode causar instabilidade ou sobrecarga de rede. Torna-se, portanto, necessário um mecanismo que integre modularidade, hierarquia e mobilidade de forma consciente.

Para endereçar esses desafios, este artigo propõe o *Mobility Impact-X* (MI-X), uma estratégia de alocação modular para *EEG Games* em névoa hierárquica. O MI-X incorpora a mobilidade no processo decisório, promovendo realocações coerentes que equilibram latência e estabilidade do sistema. As principais contribuições são:

- Estratégia de alocação modular sensível à mobilidade para *EEG Games* em arquiteturas de Névoa Hierárquica;
- Algoritmo que pondera latência, recursos e custo de comunicação para migração;
- Avaliação experimental comparativa frente a algoritmos do estado da arte.

O restante do artigo organiza-se em: Seção 2 (Trabalhos Relacionados); Seção 3 (Algoritmo Proposto); Seção 4 (Configuração Experimental); Seção 5 (Resultados); e Seção 6 (Conclusão).

## 2. Trabalhos Relacionados

Aplicações modulares e sensíveis à latência, como BCI, exigem orquestração eficiente de microsserviços distribuídos [Mahmud and Buyya 2022]. Nesse contexto, o *Computing Continuum* integra recursos da borda ao núcleo em um sistema heterogêneo e adaptável [Freire et al. 2025, Oliveira et al. 2024]. Em jogos EEG, o pipeline impõe desafios de alocação agravados pela mobilidade e flutuações de rede, exigindo gestão eficiente de clusters [Mahmud and Buyya 2022] e resiliência a falhas.

Abordagens iniciais como o DP-I [Charântola et al. 2019] priorizam módulos em *cloudlets*, mas não exploram a hierarquia multinível. Avanços posteriores consideram arquiteturas hierárquicas e múltiplos objetivos: o CB-E [Peixoto et al. 2023] foca no tráfego de rede, enquanto o LI-X [Oliveira et al. 2024] visa reduzir latência e tráfego simultaneamente. Outras estratégias utilizam meta-heurísticas para custo [Lai et al. 2020] ou focam em requisitos temporais estritos [Altin et al. 2024].

Além da latência, o Argos [Rossi et al. 2022] equilibra privacidade na borda, enquanto o Tetris [Almeida and Peixoto 2025] prioriza serviços por urgência para reduzir violações de SLA. No continuum, o SmartKV [Aznar-Poveda et al. 2024] ajusta réplicas de dados dinamicamente. Para mitigar o impacto da mobilidade, o LIP [Rac and Brorsson 2024] adapta o posicionamento por custo e o CMFogV [Araújo and Bittencourt 2024] antecipa migrações via predição. Complementarmente, o sistema DAICS [Freire et al. 2025] preserva a fluidez da informação via imputação de dados, essencial para a saúde no continuum.

A Tabela 1 sintetiza o MI-X frente aos trabalhos discutidos, destacando a interdependência modular e a adaptação dinâmica como determinantes para a Qualidade de Experiência (QoE).

**Tabela 1. Comparação dos Trabalhos Relacionados em Ambientes do Continuum**

| Autores | Estratégias | Modular | Latência | Rede | Mobilidade | Multi-level |
|---|---|---|---|---|---|---|
| [Charântola et al. 2019] | DP-I | ✓ | ✓ | ✗ | ✗ | ✗ |
| [Peixoto et al. 2023] | CB-E | ✓ | ✗ | ✓ | ✗ | ✓ |
| [Oliveira et al. 2024] | LI-X | ✓ | ✓ | ✓ | ✗ | ✓ |
| [Lai et al. 2020] | Meta-heuristic | ✓ | ✓ | ✗ | ✗ | ✓ |
| [Altin et al. 2024] | LAMOMRank | ✓ | ✓ | ✗ | ✗ | ✓ |
| [Rossi et al. 2022] | Argos | ✓ | ✓ | ✗ | ✗ | ✗ |
| [Aznar-Poveda et al. 2024] | SmartKV | ✗ | ✓ | ✗ | ✗ | ✓ |
| [Rac and Brorsson 2024] | LIP | ✗ | ✓ | ✗ | ✓ | ✓ |
| [Araújo and Bittencourt 2024] | CMFogV | ✗ | ✓ | ✗ | ✓ | ✓ |
| **Proposta** | **MI-X** | ✓ | ✓ | ✓ | ✓ | ✓ |

As estratégias mapeadas definem a sensibilidade à mobilidade como a capacidade de assegurar a adjacência entre serviço e usuário perante mudanças na topologia lógica. Todavia, soluções focadas em mobilidade negligenciam dependências entre módulos, que em BCI abrangem etapas de coleta, pré-processamento, extração de características e classificação. Ignorar tais vínculos implica em desconsiderar que o posicionamento de um componente afeta o fluxo de dados subsequente. Caso estágios dependentes sejam alocados em nós distantes, o custo de rede anula ganhos de processamento individual. O MI-X resolve esse impasse ao integrar a estrutura modular à gestão de mobilidade, mantendo componentes interdependentes próximos para preservar a baixa latência.

## 3. Mobility Impact-X

A estratégia *Mobility Impact-X* (MI-X) foi concebida para lidar com aplicações modulares sensíveis à latência executadas sobre infraestruturas de Computação em Névoa Hierárquica sujeitas à mobilidade do usuário. O objetivo central do MI-X é equilibrar, de forma dinâmica, o trade-off entre (i) proximidade computacional ao usuário, (ii) custo de comunicação induzido pela distribuição/migração de módulos e (iii) estabilidade operacional.

### 3.1. Modelagem do Sistema

Considera-se um ambiente de Computação em Névoa Hierárquica composto por um conjunto de nós computacionais organizados em múltiplos níveis, $\mathcal{F} = \mathcal{C} \cup \mathcal{N} \cup \mathcal{L}$, onde $\mathcal{C}$ representa a camada de Cloud, $\mathcal{N}$ a camada de Névoa (Fog) e $\mathcal{L}$ o conjunto de Cloudlets localizados na borda da rede. Essa organização hierárquica reflete diferenças naturais de latência, capacidade computacional e custo de comunicação entre os níveis da infraestrutura.

A hierarquia é modelada por uma função pai $\pi : \mathcal{F} \to \mathcal{F} \cup \{\emptyset\}$, em que $\pi(f)$ retorna o nó imediatamente superior a $f$ (ou $\emptyset$ se $f$ é a raiz). Para dois nós $x, y \in \mathcal{F}$, $\mathrm{LCA}(x, y)$ denota o ancestral comum mais próximo (*lowest common ancestor*). Além disso, diz-se que dois nós pertencem ao mesmo cluster hierárquico, denotado por $x \sim y$,

quando compartilham o mesmo domínio local da hierarquia (por exemplo, Cloudlets sob a mesma subárvore de Névoa), caracterizando migrações locais de menor custo.

Cada nó $f \in \mathcal{F}$ possui capacidade computacional limitada, expressa por $\mathrm{MIPS}(f)$. As aplicações são compostas por um conjunto de módulos $\mathcal{M} = \{m_1, \ldots, m_n\}$, em que cada módulo $m$ requer $\mathrm{cpu}(m)$ unidades de processamento. No instante discreto $t$, o conjunto de módulos alocados em $f$ é representado por $\mathcal{M}_f(t)$, e a carga total do nó é dada por $\mathcal{W}(f,t) = \sum_{m \in \mathcal{M}_f(t)} \mathrm{cpu}(m)$, devendo sempre satisfazer a restrição $\mathcal{W}(f,t) \leq \mathrm{MIPS}(f)$.

A execução das aplicações é orientada ao usuário final. Para cada usuário móvel $u$, define-se $g(u,t) \in \mathcal{L}$ como o Cloudlet ao qual o usuário está associado no instante $t$. Um evento de mobilidade ocorre quando há mudança nesse ponto de acesso, isto é, quando $g(u,t) \neq g(u,t{+}1)$, podendo exigir realocações dos módulos ao longo da hierarquia.

A latência fim-a-fim percebida pelo usuário no instante $t$ é denotada por $\mathcal{L}(u,t)$ e resulta da comunicação entre o usuário e os módulos da aplicação, bem como da comunicação entre módulos distribuídos na infraestrutura. Para modelar essas interações, assume-se que a aplicação define um grafo de dependências $\mathcal{G} = (\mathcal{M}, \mathcal{E})$, no qual uma aresta $(m_i, m_j) \in \mathcal{E}$ indica troca de dados entre os módulos $m_i$ e $m_j$. Assim, a latência pode ser expressa de forma abstrata como $\mathcal{L}(u,t) = \lambda_u(g(u,t)) + \sum_{(m_i,m_j)\in\mathcal{E}} w_{ij} \cdot d(f_i, f_j)$, onde $\lambda_u(\cdot)$ representa o custo dominante de comunicação entre o usuário e o Cloudlet associado, $d(f_i, f_j)$ expressa o custo de comunicação entre os nós $f_i$ e $f_j$ que hospedam os módulos $m_i$ e $m_j$, e $w_{ij}$ pondera a intensidade da comunicação entre os módulos.

Além da latência percebida, decisões de realocação de módulos impactam diretamente o tráfego da rede. Para capturar esse efeito, define-se $\kappa(m, f, t)$ como o custo incremental de comunicação associado à promoção do módulo $m$ a partir de um nó saturado $f$ no instante $t$, refletindo o aumento de tráfego ou de latência entre módulos dependentes após a realocação. De forma agregada, o custo global de comunicação no sistema pode ser representado por $\mathcal{N}(t) = \sum_{(m_i,m_j)\in\mathcal{E}} w_{ij} \cdot d(f_i, f_j)$, fornecendo uma métrica abstrata para comparar diferentes decisões de migração e seus impactos sobre a rede.

### 3.2. Comportamento do MI-X no Continuum Hierárquico

A Figura 1 demonstra o funcionamento do MI-X em uma arquitetura de Computação em Névoa Hierárquica e evidencia como o algoritmo lida com os desafios impostos por aplicações modulares sensíveis à latência sob mobilidade do usuário. A figura organiza o continuum Cloud–Névoa–Cloudlets, destacando (i) as diferenças de latência, (ii) a variação de capacidade computacional e (iii) o custo de comunicação entre camadas, e como esses fatores influenciam as decisões de alocação e migração.

Na camada de Cloudlets, localizada na borda, observa-se a menor latência de comunicação com o usuário, tornando-a o local preferencial para executar módulos críticos (ex.: etapas do pipeline que impactam diretamente a resposta percebida em EEG Games). O MI-X explora essa proximidade por meio de uma política *edgewards*, priorizando alocação no cloudlet associado ao usuário sempre que $\mathcal{W}(g(u,t),t) + \mathrm{cpu}(m) \leq \mathrm{MIPS}(g(u,t))$ for satisfeita. Entretanto, como a figura ressalta, Cloudlets operam sob forte pressão de recursos: sua capacidade limitada pode ser rapidamente saturada por

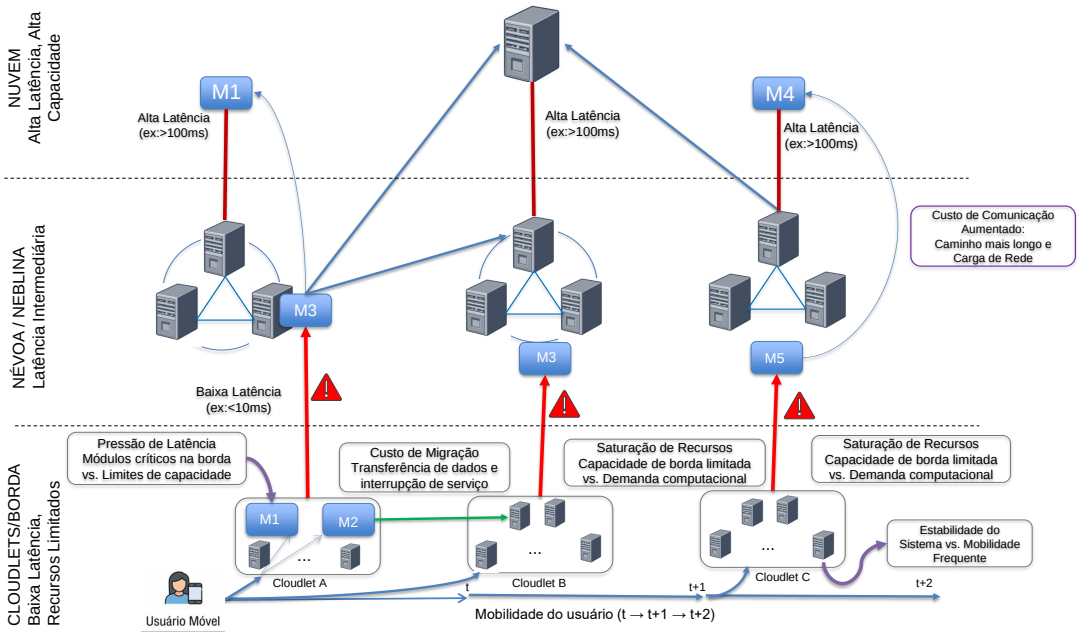



**Figura 1. Visão geral do comportamento do algoritmo MI-X**

cargas concorrentes (múltiplos usuários) ou por padrões de mobilidade que exigem realocações frequentes.

Quando ocorre saturação na borda, a figura destaca o dilema central: promover módulos para camadas superiores reduz pressão local, mas tende a aumentar latência e custo de comunicação (tanto com o usuário quanto entre módulos distribuídos). Diferentemente de estratégias que realizam essa promoção imediatamente, o MI-X introduz uma etapa intermediária de análise, representada pelos alertas associados ao custo de migração e ao impacto na rede. Em termos operacionais, isso significa que, antes de promover o novo módulo, o algoritmo avalia se remover (desalocar) um subconjunto de módulos já presentes em $f$ e promovê-los ao pai $\pi(f)$ é suficiente para liberar recursos com um impacto global menor.

A camada de Névoa desempenha um papel estrutural no MI-X ao atuar como nível intermediário, tanto em latência quanto em capacidade. A figura evidencia que a Névoa é um "amortecedor" durante mobilidade: quando o usuário se desloca entre cloudlets do mesmo cluster, a realocação pode ser direta e local. Quando a mobilidade ocorre entre clusters distintos, o MI-X promove temporariamente os módulos ao ancestral comum mais próximo, evitando migrações abruptas e saltos diretos entre extremos da hierarquia. Isso reduz instabilidades, pois preserva um caminho hierárquico coerente até que os módulos sejam reposicionados próximo ao novo $g(u, t)$.

Por fim, a camada de Nuvem (Cloud), embora dotada de alta capacidade, aparece associada a altos valores de latência e custo de comunicação. No MI-X, a nuvem é empregada apenas quando manter módulos em camadas inferiores produz um impacto maior do que promovê-los (por exemplo, em sobrecarga extrema ou quando a hierarquia intermediária já está saturada). Assim, a figura reforça que o MI-X não usa a nuvem como destino preferencial, mas como recurso complementar para absorver situações críticas.

A mobilidade do usuário, representada ao longo dos instantes $t$, $t{+}1$ e $t{+}2$, intensifica os desafios de alocação ao exigir realocações frequentes. A figura associa esses eventos ao aumento do custo de migração e ao risco de instabilidade, explicitando o trade-off entre manter baixa latência e preservar estabilidade operacional. Nesse sentido, o MI-X assume conscientemente o custo associado à mobilidade quando isso reduz $\mathcal{L}(u,t)$, mas utiliza a análise de impacto para evitar decisões que causem aumentos desproporcionais em $\mathcal{N}(t)$.

Em síntese, a Figura 1 delimita o espaço de decisão explorado pelo MI-X e evidencia como o algoritmo integra mobilidade, hierarquia e análise de impacto de comunicação para balancear latência, uso de recursos e estabilidade. Ao privilegiar execução na borda sempre que possível e recorrer à promoção hierárquica apenas quando estritamente necessário e com menor impacto, o MI-X suporta eficientemente aplicações modulares sensíveis à latência em ambientes dinâmicos.

### 3.3. Processo de Alocação e Migração

Para tornar o comportamento mais concreto, a Figura 2 apresenta um exemplo ilustrativo do processo de alocação e migração adotado pelo MI-X. O exemplo evidencia como os módulos acompanham o deslocamento do usuário ao longo da hierarquia Cloudlet–Névoa–Cloud, preservando a execução na borda sempre que viável e recorrendo à promoção hierárquica somente quando a restrição de capacidade impedir a execução local com estabilidade.

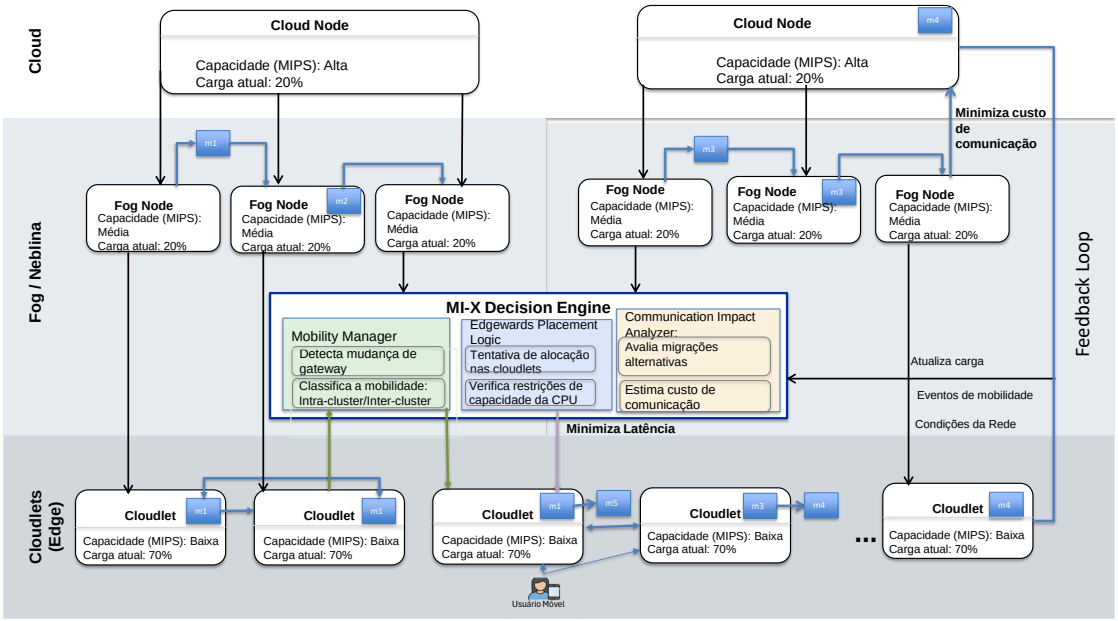



**Figura 2. Exemplo ilustrativo do processo de alocação e migração do MI-X**

O *MI-X Decision Engine* atua como o núcleo do processo decisório, integrando três funções: (i) detecção de mobilidade (mudanças em $g(u,t)$), (ii) lógica *edgewards* (tentativa de manter módulos próximos ao usuário) e (iii) análise de impacto (decisão sobre quais módulos promover quando ocorre saturação). Assim, quando o usuário muda de gateway, o algoritmo executa uma realocação controlada; quando há saturação, ele

decide entre promover o novo módulo ou desalocar/promover módulos existentes, sempre visando minimizar o aumento de custo de comunicação estimado.

### 3.4. Gerenciamento de Mobilidade

A modelagem do MI-X fundamenta-se em uma estrutura hierárquica de grafos e funções de custo, cujas principais notações e variáveis estão sumarizadas na Tabela 2. O sistema é representado por um conjunto de nós distribuídos entre Nuvem (C), Névoa (N) e Borda (L), onde a topologia é regida pela função de ancestralidade pi(f) e pela identificação do ancestral comum mais próximo, LCA(x,y).

**Tabela 2. Notação e variáveis do modelo MI-X.**

| Símbolo | Descrição |
|---|---|
| $\mathcal{C}, \mathcal{N}, \mathcal{L}$ | Conjuntos de nós da hierarquia: Nuvem, Névoa e Borda/Cloudlets. |
| $\pi(f), LCA(x, y)$ | Funções de hierarquia: nó pai de $f$ e ancestral comum mais próximo. |
| $MIPS(f), \mathcal{W}(f, t)$ | Capacidade total e carga computacional do nó $f$ no instante $t$. |
| $\mathcal{M}, cpu(m)$ | Módulos da aplicação e suas respectivas demandas de processamento. |
| $\mathcal{M}_f(t), g(u, t)$ | Módulos alocados no nó $f$ e ponto de acesso do usuário $u$. |
| $\mathcal{L}(u, t), \mathcal{N}(t)$ | Latência percebida pelo usuário e custo global de rede do sistema. |
| $d(f_i, f_j), w_{ij}$ | Custo de distância entre nós e intensidade da comunicação (Grafo $\mathcal{G}$). |
| $\alpha, \mathcal{S}^*, \Delta, \omega$ | Parâmetros de decisão: ancestral ($\alpha$), candidatos ($\mathcal{S}^*$), déficit ($\Delta$) e impacto ($\omega$). |

O gerenciamento de mobilidade traduz eventos de deslocamento do usuário em decisões consistentes de realocação de módulos ao longo da hierarquia. O Algoritmo 1 recebe o usuário $u$, o novo gateway $g'$ e o instante $t$. Se $g(u, t) = g'$, não há mobilidade e o algoritmo encerra. Caso contrário, define-se $\mathcal{M}_u$ como o conjunto de módulos associados ao usuário (por exemplo, módulos da aplicação instanciados para $u$). Em seguida, o algoritmo diferencia dois cenários: (i) mobilidade intra-cluster, na qual os módulos são realocados diretamente para $g'$, e (ii) mobilidade inter-cluster, na qual os módulos são promovidos ao ancestral comum $\alpha = \text{LCA}(g, g')$ antes de serem realocados ao destino final. Essa migração em duas etapas reduz instabilidades e evita oscilações decorrentes de saltos hierárquicos longos.

**Algoritmo 1** MI-X: Gerenciamento de Mobilidade

**Require:** usuário $u$, novo gateway $g'$, instante $t$
$g \leftarrow g(u, t)$
**if** $g = g'$ **then return**
**end if**
$\mathcal{M}_u \leftarrow \{m \mid m \text{ associado a } u\}$
**if** $g \sim g'$ **then**
  **for all** $m \in \mathcal{M}_u$ **do**
    $\mathcal{M}_g(t) \leftarrow \mathcal{M}_g(t) \setminus \{m\}$
    ALLOCATE$(m, g', t)$
  **end for**
**else**
  $\alpha \leftarrow \text{LCA}(g, g')$
  **for all** $m \in \mathcal{M}_u$ **do**
    $\mathcal{M}_g(t) \leftarrow \mathcal{M}_g(t) \setminus \{m\}$
    ALLOCATE$(m, \alpha, t)$
    $\mathcal{M}_\alpha(t) \leftarrow \mathcal{M}_\alpha(t) \setminus \{m\}$
    ALLOCATE$(m, g', t)$
  **end for**
**end if**
$g(u, t{+}1) \leftarrow g'$

### 3.5. Alocação Hierárquica e Seleção por Impacto

O Algoritmo 2 formaliza a rotina ALLOCATE utilizada tanto em eventos de mobilidade quanto em eventos de saturação. Dado um módulo $m$ e um nó destino $f$, o algoritmo tenta primeiro uma alocação direta (linhas 1–4), verificando se a restrição de capacidade é satisfeita. Caso contrário, o MI-X considera o nó pai $p = \pi(f)$. Se não houver pai (raiz), a alocação falha.

**Algoritmo 2** MI-X: Alocação Hierárquica

**Require:** módulo $m$, nó destino $f$, instante $t$
1: **if** $\mathcal{W}(f,t) + \mathrm{cpu}(m) \leq \mathrm{MIPS}(f)$ **then**
2: $\quad \mathcal{M}_f(t) \leftarrow \mathcal{M}_f(t) \cup \{m\}$ **return**
3: **end if**
4: $p \leftarrow \pi(f)$
5: **if** $p = \emptyset$ **then return**
6: **end if**
7: $\mathcal{S}^\star \leftarrow$ IMPACT$(m, f, t)$
8: **if** $m \in \mathcal{S}^\star$ **then**
9: $\quad$ ALLOCATE$(m, p, t)$ **return**
10: **end if**
11: $\Delta \leftarrow \mathcal{W}(f,t) + \mathrm{cpu}(m) - \mathrm{MIPS}(f)$
12: $\mathcal{E} \leftarrow$ SELECTEVICT$(\mathcal{S}^\star \setminus \{m\}, \Delta)$
13: **for all** $m_k \in \mathcal{E}$ **do**
14: $\quad \mathcal{M}_f(t) \leftarrow \mathcal{M}_f(t) \setminus \{m_k\}$
15: $\quad$ ALLOCATE$(m_k, p, t)$
16: **end for**
17: ALLOCATE$(m, f, t)$

O passo central é a chamada IMPACT (linha 7), que retorna um conjunto $\mathcal{S}^\star$ de módulos candidatos cuja promoção minimiza o impacto de comunicação estimado. Se $m \in \mathcal{S}^\star$, então o algoritmo decide promover o próprio módulo ao pai (linhas 8–11), evitando desalocar outros módulos. Caso contrário, significa que é mais vantajoso desalocar/promover um subconjunto de módulos já alocados. Para isso, calcula-se o déficit de capacidade $\Delta$ (linha 12) e seleciona-se um conjunto de remoção $\mathcal{E}$ (linha 13) que libera pelo menos $\Delta$ de CPU. Esses módulos são removidos de $f$ e promovidos para $p$ (linhas 14–17). Por fim, com recursos liberados, o módulo $m$ é alocado em $f$ (linha 18).

O Algoritmo 3 detalha a função IMPACT. Para um nó $f$ considerado saturado, define-se $\mathcal{A} = \mathcal{M}_f(t) \cup \{m\}$ como o conjunto de módulos em disputa (os já alocados mais o novo). A função CANDIDATES$(\mathcal{A})$ gera subconjuntos candidatos $\mathbb{C}$ (por exemplo, limitando a cardinalidade para evitar explosão combinatória). Para cada subconjunto $\mathcal{S} \in \mathbb{C}$, calcula-se o custo agregado $\omega = \sum_{x \in \mathcal{S}} \kappa(x, f, t)$, isto é, o impacto estimado na rede caso os elementos de $\mathcal{S}$ sejam promovidos ao pai de $f$. O subconjunto com menor custo é retornado como $\mathcal{S}^\star$ e usado pelo Algoritmo 2 para decidir entre promover $m$ ou promover módulos existentes.

Dessa forma, o MI-X integra mobilidade, alocação hierárquica e análise de impacto em um ciclo contínuo de decisão, mantendo o custo de comunicação controlado e proporcionando estabilidade operacional em cenários dinâmicos.

## 4. Organização e Configuração dos Experimentos

Os experimentos foram conduzidos no *iFogSim 2*, considerando a definição da aplicação, uma topologia hierárquica Cloud–Fog–Cloudlets e a modelagem da mobilidade do usuário.

**Algoritmo 3** MI-X: Análise de Impacto de Comunicação

**Require:** novo módulo $m$, nó saturado $f$, instante $t$
$\mathcal{A} \leftarrow \mathcal{M}_f(t) \cup \{m\}$
$\mathbb{C} \leftarrow$ CANDIDATES$(\mathcal{A})$
$\omega^\star \leftarrow +\infty, \mathcal{S}^\star \leftarrow \emptyset$
**for all** $\mathcal{S} \in \mathbb{C}$ **do**
  $\omega \leftarrow \sum_{x \in \mathcal{S}} \kappa(x, f, t)$
  **if** $\omega < \omega^\star$ **then**
    $\omega^\star \leftarrow \omega$
    $\mathcal{S}^\star \leftarrow \mathcal{S}$
  **end if**
**end forreturn** $\mathcal{S}^\star$

### 4.1. Aplicação: EEG Game

A aplicação simulada é um EEG Game, representando uma Interface Cérebro–Computador em tempo real. Embora avaliado nesse cenário, o problema investigado pertence à classe de aplicações BCI sensíveis à latência, sendo o EEG Game adotado como uma instância representativa, e não restritiva. Um usuário (jogador) utiliza um dispositivo de EEG vestível, e os sinais cerebrais são processados para acionar respostas no jogo. Trata-se de uma aplicação modular e crítica à latência, na qual o atraso entre a intenção do usuário (sinal de EEG) e o feedback visual degrada diretamente a experiência [Xu et al. 2023]. Além disso, o fluxo de dados é intensivo, o que torna o posicionamento dos módulos e o custo de comunicação determinantes para o desempenho.

Para representar a mobilidade do usuário, a simulação baseou-se em um *dataset* do iFogSim 2 associado a um mapa urbano da região de Melbourne, Austrália (Figura 3), cujas coordenadas geográficas permitem instanciar trajetórias móveis realistas.

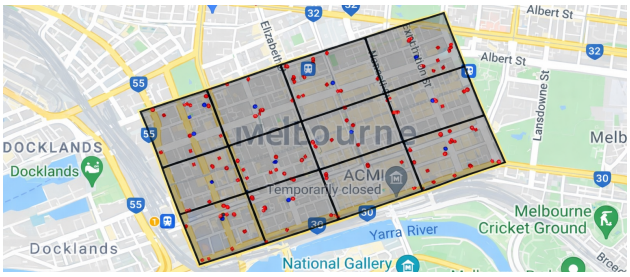
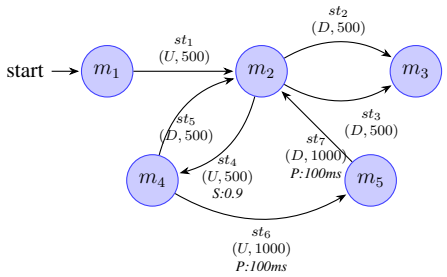



**Figura 3. Área urbana da região de Melbourne, Austrália, considerada nos experimentos. Adaptado de [Mahmud and Buyya 2022]**

A partir das coordenadas sequenciais, adotou-se um modelo de mobilidade acíclico e com velocidade fixa (*DIRECTIONAL MOBILITY*), permitindo avaliar, de forma realista, o efeito do deslocamento do usuário sobre a alocação/migração de módulos em nós distribuídos, o que é um aspecto crítico em EEG Games, que exigem baixa latência e continuidade de serviço durante mudanças de ponto de acesso; essa escolha é consistente com trabalhos que também utilizam o iFogSim [Mahmud and Buyya 2022, Charântola et al. 2019]. A Figura 4 sintetiza a aplicação modular e suas dependências: o EEG Game segue um pipeline com *Pre-processing*, *Feature Extraction* e *Classification*, modelado no *iFogSim 2* por *AppModule* (requisitos de processamento) e *AppEdge* (dependências/fluxos), incluindo periodicidade, seletividade e tamanhos de dados conforme o grafo.




**(a) Topologia**

| Index | Component |
|---|---|
| $m_1$ | EEG Sensor |
| $m_2$ | Client |
| $m_3$ | Display |
| $m_4$ | Calc. |
| $m_5$ | Coord. |

**(b) Componentes**

| Arc | Action |
|---|---|
| $st_1$ | EEG |
| $st_2$ | Self state update |
| $st_3$ | Global state update |
| $st_4$ | Sensor |
| $st_5$ | Concentration |
| $st_6$ | Player game state |
| $st_7$ | Global game state |

**(c) Transições**

**Figura 4. Diagrama do EEGTBG: (a) topologia, (b) componentes e (c) transições de estado.**

### 4.2. Topologia de Simulação e Algoritmos Comparados

A avaliação considera os algoritmos Cloud-Based Extreme (CB-E), Dynamic Placement-Instance (DP-I), Latency-Improved-X (LI-X) e a proposta Mobility-aware Inter-fog eXchange (MI-X). Os experimentos utilizam a topologia hierárquica ilustrada na Figura 5 composta por nuvem, gateway e *cloudlets* com o dispositivo de Interface Cérebro-Computador (BCI, do inglês *Brain-Computer Interface*) na camada final. Os atrasos entre camadas são parametrizados conforme a figura para quantificar o efeito de decisões de *offloading*, definido como a transferência de tarefas para infraestruturas remotas, e de migração ao longo do *continuum*. Essa gradação permite avaliar a sensibilidade da aplicação quanto à distância física do processamento e à disponibilidade de recursos computacionais.

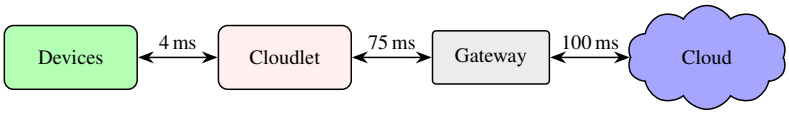



**Figura 5. Topologia de simulação**

A definição dessas latências (4,ms, 75,ms e 100,ms) fundamenta-se na segmentação física da rede: o atraso de 4,ms reflete a comunicação de salto único (one-hop) via rádio na borda; os 75,ms simulam o tráfego em infraestruturas metropolitanas (MAN) entre nós de névoa; e os 100,ms representam o acesso a centros de dados remotos via WAN. Essa gradação é essencial para avaliar a sensibilidade da aplicação de BCI à distância física do processamento, permitindo que o MI-X explore o trade-off entre a localidade de dados e a disponibilidade de recursos computacionais.

## 5. Resultados Experimentais

Esta seção apresenta os resultados comparativos entre MI-X, CB-E, DP-I e LI-X, com foco em métricas diretamente relacionadas a aplicações BCI em tempo real: padrão de alocação de módulos, latência média, estabilidade (MTTR) e consumo de rede.

### 5.1. Padrão de Alocação de Módulos e Implicações de Latência

A Figura 6 compara o padrão de alocação de módulos entre MI-X, LI-X, CB-E e DP-I. Observa-se que o MI-X (Figura 6a) mantém a maior parte dos módulos nos *cloudlets*

(barras azuis) mesmo sob aumento de carga, favorecendo a execução próxima ao usuário. Em contraste, o LI-X (Figura 6b) intensifica o *offloading* para a nuvem (barras vermelhas) quando há saturação local, comportamento alinhado à sua orientação de reduzir pressão local e tráfego em segmentos específicos, conforme discutido em [Oliveira et al. 2024]. As abordagens CB-E (Figura 6c) e DP-I (Figura 6d) exibem transições mais abruptas de alocação: em especial, o DP-I promove módulos progressivamente para a nuvem à medida que a carga cresce, enquanto o CB-E alterna entre borda e nuvem em faixas específicas de usuários. Em aplicações estritamente interativas, tais padrões são críticos porque deslocar processamento para níveis superiores tende a aumentar a latência fim-a-fim devido ao maior atraso de propagação.

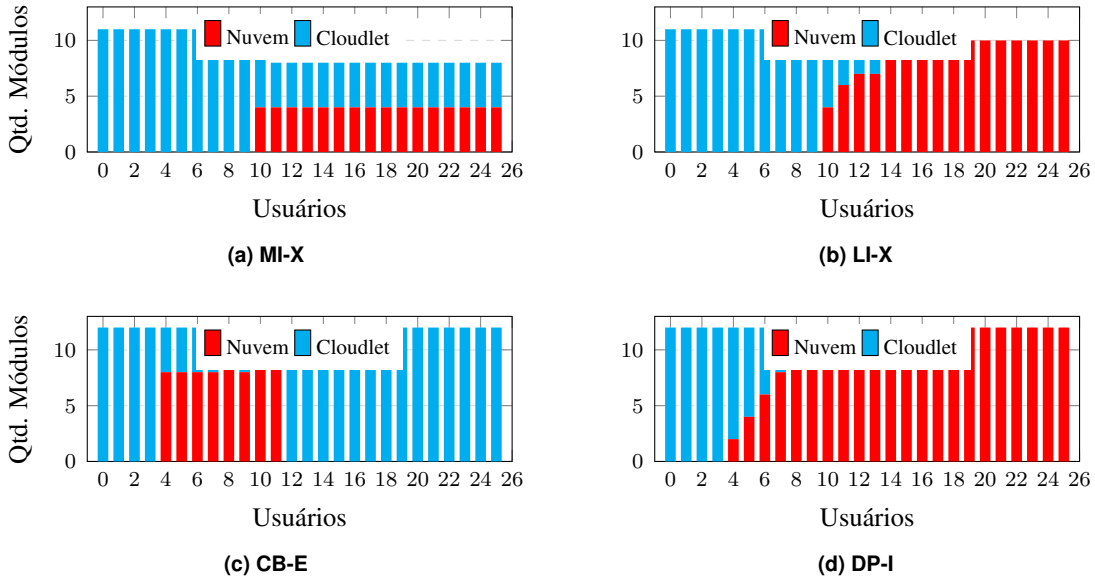


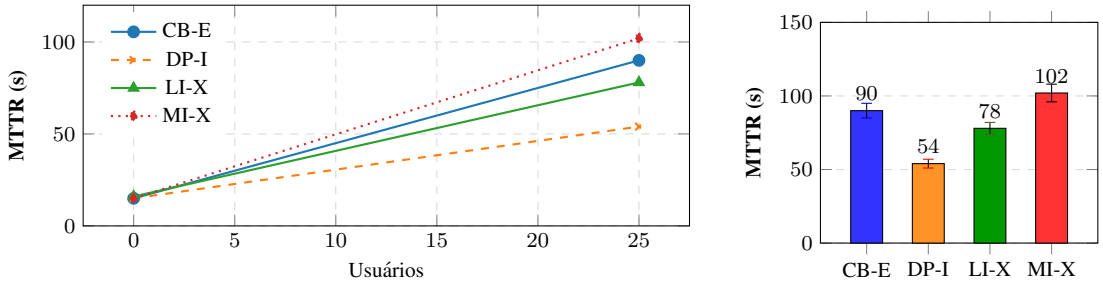

**Figura 6. Comparação do padrão de alocação de módulos.**

Esse padrão de alocação se reflete diretamente na latência média (Figura **??**(b)). O MI-X obtém a menor latência (158 ms), atendendo ao requisito de interatividade do EEG Game e superando CB-E (568 ms), DP-I (743 ms) e LI-X (746 ms). Em termos operacionais, o resultado indica que manter módulos próximos ao usuário é decisivo quando o atraso de propagação para níveis superiores domina a latência fim-a-fim.

### 5.2. Estabilidade e Custo Operacional da Mobilidade (MTTR)

A Figura 7 apresenta o comportamento do Tempo Médio para Reparo (MTTR) sob diferentes níveis de carga, bem como sua média agregada entre os cenários experimentais. Enquanto o gráfico de linhas permite observar a trajetória de recuperação à medida que a pressão do sistema aumenta, o gráfico de barras sintetiza o custo operacional associado a cada estratégia de alocação.

Observa-se que o MI-X registra o maior MTTR médio ($\approx$102 s), superando CB-E (90 s), LI-X (78 s) e DP-I (54 s). Esse resultado, contudo, não deve ser interpretado como perda de eficiência, mas como consequência direta de uma decisão arquitetural: priorizar a execução próxima ao usuário mesmo diante de mobilidade intensa. Diferentemente de abordagens reativas que evitam migrações para reduzir indisponibilidades momentâneas, o MI-X realiza transferências de contexto e reconfigurações com maior rigor, preservando



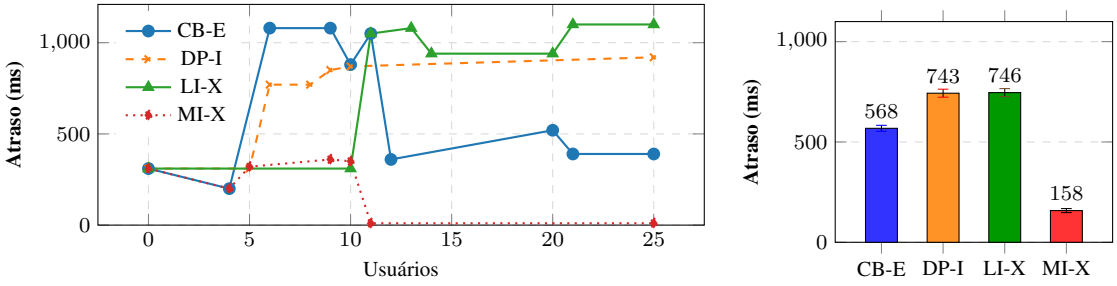

**(a) MTTR em função da carga de usuários** **(b) MTTR médio global**

**Figura 7. MTTR: comportamento sob aumento de carga.**

a localidade do serviço. Assim, o aumento no MTTR representa um custo operacional previsível do suporte à mobilidade, o que significa um investimento necessário para impedir degradações mais severas de desempenho. Esse trade-off torna-se particularmente relevante quando analisado sob a perspectiva da experiência do usuário, pois a estabilidade operacional isolada não captura integralmente os efeitos percebidos na comunicação. Dessa forma, para compreender se esse custo resulta em benefícios sistêmicos, torna-se necessário examinar seu impacto direto na latência fim a fim.

### 5.3. Latência sob Pressão de Carga

A Figura 8 ilustra permite avaliar tanto a resiliência dinâmica quanto a capacidade das estratégias de manter níveis aceitáveis de qualidade de serviço sob saturação progressiva dos recursos.



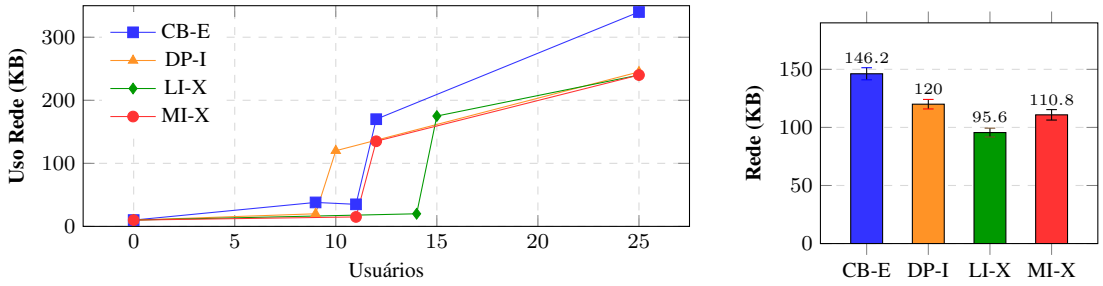

**(a) Atraso fim a fim em função da carga** **(b) Atraso médio global**

**Figura 8. Atraso: resposta dinâmica ao aumento de usuários.**

Os resultados demonstram a superioridade do MI-X, que atinge o menor atraso médio (158 ms) frente aos elevados índices do CB-E (568 ms), DP-I (743 ms) e LI-X (746 ms). Notavelmente, após o 11º usuário, o atraso do MI-X converge para valores mínimos ( 10 ms), enquanto os demais estabilizam em patamares proibitivos. Essa estabilidade decorre da maturidade da política Inter-fog, que realiza o handoff e garante o processamento no nó de borda (Tier 1) mais próximo ao atingir uma densidade crítica de recursos. Assim, o overhead de sinalização compensa-se ao evitar camadas superiores (Cloud), limitando a latência ao tempo de propagação de um único salto. O MI-X desloca o sistema para uma região de operação mais eficiente, mitigando o efeito ”ping-pong”e garantindo estabilidade sob carga. Contudo, essa redução drástica da latência implica custos operacionais, como o aumento na atividade de comunicação e no volume de dados transferidos.

### 5.4. Eficiência de Comunicação e Sustentabilidade Operacional

A Figura 9 apresenta o comportamento do consumo de rede em função da carga e sua média agregada.




**(a) Consumo de rede por usuário** **(b) Tráfego médio global**

**Figura 9. Rede: comportamento do tráfego sob aumento de usuários.**

Embora o MI-X não seja o algoritmo de menor tráfego, posição ocupada pelo LI-X (95.56 KB), seu consumo moderado (110.82 KB) revela uma estratégia substancialmente mais eficiente que a do CB-E, cuja abordagem reativa gera sobrecarga significativa (146.18 KB). Esse padrão sugere que o MI-X não busca minimizar o tráfego de forma absoluta, mas sim qualificá-lo: migrações são realizadas de maneira mais criteriosa, privilegiando nós com maior impacto sistêmico e reduzindo realocações desnecessárias. O resultado é um equilíbrio operacional no qual um aumento controlado do tráfego viabiliza ganhos expressivos em latência sem provocar pressão excessiva sobre a infraestrutura.

Em conjunto, os resultados das Figuras 7–9 mostram que a eficiência em ambientes dinâmicos depende da coordenação entre estabilidade, comunicação e proximidade computacional, e não da otimização isolada de métricas. Ao internalizar o custo da mobilidade, o MI-X conduz o sistema a um regime mais previsível e responsivo, adequado a aplicações sensíveis à latência no contínuo Edge–Cloud.

## 6. Conclusão

Este trabalho demonstra que tratar a mobilidade como variável estrutural é fundamental para aplicações BCI sensíveis à latência no continuum Edge–Cloud. Através de alocação hierárquica e análise de impacto, o MI-X otimiza o sistema ao priorizar a proximidade computacional e a estabilidade global. Os resultados indicam uma redução de até 79% no atraso médio frente aos baselines, assegurando interatividade em tempo real sob alta mobilidade. Esse ganho implica um aumento controlado de até 31% no MTTR (em relação ao LI-X), custo operacional necessário para preservar a localidade do serviço. Contudo, o consumo de rede manteve-se 24% inferior ao CB-E, reforçando que a eficiência do continuum resulta do equilíbrio entre comunicação, estabilidade e proximidade. Pesquisas futuras focarão em mecanismos preditivos para antecipar a mobilidade, reduzindo custos de migração e ampliando a responsividade.

## Disponibilidade de Artefatos

Em aderência aos princípios da Ciência Aberta, o código-fonte e o dataset utilizados neste trabalho podem ser acessados em: https://github.com/RodrigoAB93/mi-x_sbrc2026.

## Agradecimentos

Este estudo foi financiado, em parte, pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, pela FAPESB, e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brasil, sob a concessão nº 403231/2023-0.

## Referências

Almeida, L. and Peixoto, M. (2025). Tetris: An sla-aware application placement strategy. *arXiv:2511.00294*.

Altin, L., Topcuoglu, H. R., and Gürgen, F. S. (2024). Latency-aware multi-objective fog scheduling. *IEEE Access*, 12:62543–62557.

Araújo, M. C. and Bittencourt, L. F. (2024). Cmfogv: Proactive content migration. *Pervasive Mob. Comput.*, 102.

Aznar-Poveda, J. et al. (2024). Sdkv: Smart and distributed key-value store. In *Proc. SEC*, pages 1–8.

Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., and Parashar, M. (2017). Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput.*, 4:26–35.

Bittencourt, L. F. and Oliveira, R. (2017). *The IoT, Fog and Cloud Continuum*. Springer.

Charântola, D. et al. (2019). Component-based scheduling for fog computing. In *Proc. UCC*, pages 3–8.

Freire, M. et al. (2025). Clear data, clear roads: Imputing missing data. *SSRN Electronic Journal*.

Lai, P. et al. (2020). Cost-effective app user allocation. *IEEE Trans. Cloud Comput.*

Li, S., Xu, L. D., and Zhao, S. (2015). The internet of things: a survey. *Inf. Syst. Front.*, 17:243–259.

Mahmud, R. and Buyya, R. (2022). ifogsim2: Extended simulator for mobility. *Softw. Pract. Exp.*, 52(7):1525–1545.

Oliveira, L. T., Bittencourt, L. F., Genez, T. A., de Lara, E., and Peixoto, M. L. M. (2024). Enhancing modular application placement. *Comput. Commun.*, 216:95–111.

Peixoto, M. L. M. et al. (2023). Fogjam: Detecting traffic congestion in vanet. *Ad Hoc Networks*, 140:103046.

Peixoto, M. L. M., Genez, T. A., and Bittencourt, L. F. (2021). Hierarchical scheduling in multi-level fog computing. *IEEE Trans. Serv. Comput.*, 15(5):2824–2837.

Rac, S. and Brorsson, M. (2024). Cost-aware service placement and scheduling. *ACM Trans. Archit. Code Optim.*, 21.

Rossi, F., Souza, P., et al. (2022). Latency-aware privacy-preserving service migration. In *Proc. CLOSER*.

Xu, X., Liu, Y., and Zhang, X. (2023). Latency-aware offloading in mec for vr-based bci. *Future Gener. Comput. Syst.*, 145:123–134.