Avaliação do Desempenho e da Confiabilidade da Comunicação em Sistemas IoT Médicos com Arquitetura Multi-Cloud

João Tejo[1], **Thiago Bezerra** [1], **Gustavo Callou**[2], **Eduardo Tavares**[1]

[1]Centro de Informática,
Universidade Federal de Pernambuco, Recife, Brasil.

[2]Departamento de Computação,
Universidade Federal Rural de Pernambuco, Recife, Brasil

{jlvat,tvb,eagt}@cin.ufpe.br, gustavo.callou@ufrpe.br

Abstract. *The growing use of connected devices in patient monitoring requires communication architectures that provide low delay, high availability, and service continuity, especially under adverse conditions. However, experimental evaluations addressing the joint behavior of communication performance and reliability in multi-cloud IoT architectures under realistic fault scenarios remain limited. This work presents an experimental evaluation of communication performance and reliability in a distributed IoT architecture applied to a healthcare monitoring scenario, based on multi-cloud redundancy, data replication, and automatic failover between cloud providers, using embedded devices and MQTT-based transmission. The analysis considers metrics such as latency, jitter, availability, packet delivery rate, failover time, and traffic distribution between servers. The results show the communication behavior under both normal conditions and injected faults, as well as the effect of inter-provider redundancy on the continuity of data publication.*

Resumo. *O uso crescente de dispositivos conectados no monitoramento de pacientes requer arquiteturas de comunicação que ofereçam baixo atraso, alta disponibilidade e continuidade do serviço, especialmente sob condições adversas. Contudo, avaliações experimentais que analisem conjuntamente o desempenho e a confiabilidade da comunicação em arquiteturas IoT multi-cloud sob cenários realistas de falha ainda são limitadas. Este trabalho apresenta uma avaliação experimental do desempenho e da confiabilidade da comunicação em uma arquitetura IoT distribuída, aplicada a um cenário de monitoramento em saúde, baseada em multi-cloud, com redundância ativa, replicação de dados e failover automático entre provedores de nuvem, utilizando dispositivos embarcados e transmissão via MQTT. A análise considera métricas como latência, jitter, disponibilidade, taxa de entrega de pacotes, tempo de failover e distribuição do tráfego entre servidores. Os resultados mostram o comportamento da comunicação em condições normais e sob falhas injetadas, bem como o efeito da redundância entre provedores sobre a continuidade da publicação de dados.*

1. Introdução

A Internet das Coisas (IoT) tem sido cada vez mais aplicada ao monitoramento contínuo de pacientes em ambientes clínicos e remotos, integrando sensores, dispositivos embarcados, redes de comunicação e serviços em nuvem. Em aplicações voltadas à saúde, a comunicação de dados precisa ocorrer com regularidade e dentro de limites aceitáveis de atraso, pois interrupções, perdas de mensagens ou degradações prolongadas podem comprometer a continuidade do monitoramento e o funcionamento do sistema [Bezerra et al. 2025].

Embora existam estudos sobre comunicação em sistemas IoT para saúde, muitas avaliações são conduzidas em condições controladas, com foco em métricas tradicionais de desempenho e menor atenção ao comportamento do sistema diante de falhas. Em particular, permanece pouco explorada uma avaliação experimental que integre comunicação MQTT, arquitetura multi-cloud com redundância entre provedores, replicação de dados e análise conjunta de métricas de desempenho e confiabilidade sob falhas controladas na infraestrutura [Nguyen et al. 2021].

Nesse contexto, este artigo apresenta uma avaliação experimental do desempenho e da confiabilidade da comunicação em uma arquitetura IoT distribuída, aplicada a um cenário de monitoramento em saúde, baseada em multi-cloud, com redundância ativa, replicação de dados e *failover* automático entre provedores de nuvem. A solução utiliza dispositivos embarcados e transmissão via *Message Queuing Telemetry Transport* (MQTT), permitindo observar o comportamento do sistema tanto em condições normais quanto em situações de instabilidade. O ambiente experimental emprega um sensor de temperatura e umidade (DHT22) para a geração periódica de dados, cujo padrão de publicação MQTT é compatível com cenários de monitoramento remoto, sendo o foco da avaliação o comportamento da infraestrutura de comunicação e não o tipo de dado coletado. A análise considera métricas de desempenho e confiabilidade, incluindo latência, jitter, disponibilidade, taxa de entrega de pacotes, tempo de *failover* e distribuição do tráfego entre servidores. Os resultados são apresentados por meio de quatro estudos de caso que analisam diferentes dimensões do comportamento da arquitetura proposta em condições normais de operação, sob falhas e após a recuperação do sistema.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 discute os conceitos teóricos utilizados. A Seção 4 descreve a metodologia experimental. A Seção 5 apresenta a arquitetura proposta. A Seção 6 apresenta os estudos de caso e a discussão dos resultados. Por fim, a Seção 7 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Lakhan et al. [Lakhan et al. 2023] propõem um framework tolerante a falhas para IoT industrial em saúde baseado em gêmeo digital (*digital twin*) e redes fog-cloud federadas, no qual tarefas computacionais são distribuídas entre nós de fog e múltiplos provedores de nuvem com mecanismo de *failover* automático. A avaliação considera latência como principal métrica, mas não adota injeção de falhas controlada nem inclui métricas como jitter, taxa de entrega de pacotes ou tempo de recuperação. Cao et al. [Cao et al. 2019] apresentam uma arquitetura multi-cloud escalável para IoT médica, em que dados clínicos são distribuídos e replicados entre provedores distintos para favorecer disponibilidade

e escalabilidade de armazenamento. A análise considera tempo de resposta, mas não inclui *failover* automático no nível do dispositivo nem cenários com falhas induzidas para observar o comportamento do sistema sob interrupções.

Upadhyay et al. [Upadhyay et al. 2023] analisam desafios e limitações de sistemas IoT para saúde inteligente, comparando protocolos como MQTT, *Constrained Application Protocol* (CoAP) e *Hypertext Transfer Protocol* (HTTP) quanto à adequação em cenários com dispositivos de recursos limitados. O estudo considera latência de transmissão, mas não aborda redundância entre provedores de nuvem nem mecanismos de tolerância a falhas na infraestrutura. Wadullah e Khaleel [Wadullah Tareq and Ahmed Khaleel 2021] estudam a implementação do protocolo MQTT em sistemas de saúde baseados em IoT, avaliando o desempenho da comunicação em termos de latência e taxa de entrega de pacotes (PDR) em diferentes níveis de qualidade de serviço (QoS). O foco permanece na comunicação, sem considerar arquiteturas distribuídas com replicação de dados ou comutação entre servidores. Stergiou et al. [Stergiou et al. 2023] investigam a transmissão segura de dados IoT em cenário de gêmeo digital para saúde, propondo replicação entre camadas de nuvem com foco em confidencialidade. A avaliação utiliza vazão como métrica, sem considerar redundância ativa entre provedores ou *failover*. Sornlertlamvanich et al. [Sornlertlamvanich et al. 2025] propõem uma arquitetura de telemonitoramento remoto integrando IoT, gateway e nuvem com segurança via HTTPS/QUIC, avaliando a latência de transmissão com e sem criptografia em um protótipo com Raspberry Pi 5 e sensores de ECG e SpO_2, sem considerar redundância entre provedores ou mecanismos de *failover*.

Bezerra et al. [Bezerra et al. 2025] propõem uma abordagem baseada em modelagem para avaliação conjunta de desempenho e disponibilidade em sistemas IoMT, utilizando Redes de Petri Estocásticas para analisar o impacto de mecanismos de redundância sobre o tempo de resposta e a indisponibilidade do sistema. Os resultados indicam que a redundância pode reduzir o tempo de indisponibilidade sem degradar o desempenho. Entretanto, o estudo é centrado em modelagem e não considera uma avaliação experimental com injeção de falhas em uma arquitetura MQTT distribuída entre provedores de nuvem. Nguyen et al. [Nguyen et al. 2021] apresentam uma metodologia hierárquica para quantificar dependability e segurança em uma infraestrutura IoMT baseada em cloud–fog–edge, combinando *Fault Trees* e cadeias de Markov em tempo contínuo. A análise considera métricas como confiabilidade ao longo do tempo e disponibilidade em regime permanente, além de investigar a sensibilidade da arquitetura a parâmetros de falha, recuperação e ataques cibernéticos. Embora o trabalho fortaleça a análise formal de confiabilidade e tolerância a falhas em arquiteturas IoMT, não enfoca métricas de comunicação como latência, jitter, taxa de entrega de pacotes e distribuição de tráfego em um ambiente experimental multi-cloud.

A Tabela 1 sintetiza a comparação entre os trabalhos segundo cinco critérios: uso do protocolo MQTT, arquitetura multi-cloud, redundância ativa com *failover* entre provedores, injeção controlada de falhas na infraestrutura e avaliação experimental com *testbed* físico. Observa-se que nenhum dos trabalhos analisados reúne simultaneamente os cinco critérios. Alguns concentram-se em métricas pontuais de comunicação sem considerar redundância entre provedores, enquanto outros priorizam disponibilidade e tolerância a falhas com ênfase em modelagem analítica. Este trabalho reúne esses cinco aspectos em

um mesmo cenário experimental.

Tabela 1. Comparação entre os trabalhos relacionados.

Trabalho	MQTT	Multi-cloud	Red. ativa	Inj. falhas	Aval. exp.
Lakhan et al.		✓	✓		
Cao et al.		✓			✓
Upadhyay et al.	✓				
Wadullah e Khaleel	✓				✓
Stergiou et al.		✓			✓
Sornlertl. et al.					✓
Bezerra et al.			✓		
Nguyen et al.			✓		
Este trabalho	✓	✓	✓	✓	✓

3. Referencial Teórico

3.1. IoT Aplicada à Saúde

A aplicação da Internet das Coisas (IoT) ao monitoramento de pacientes em ambientes clínicos e remotos envolve a integração de dispositivos conectados, sensores e serviços computacionais. Nesse contexto, a comunicação entre dispositivos, infraestrutura de rede e serviços de processamento torna-se parte essencial do funcionamento da aplicação [Bezerra et al. 2025]. Por operar em cenários sensíveis, tais soluções demandam não apenas conectividade, mas também disponibilidade, confiabilidade, tolerância a falhas e capacidade de recuperação diante de eventos adversos [Nguyen et al. 2021].

3.2. Métricas de Avaliação

A avaliação de sistemas IoT distribuídos requer métricas que permitam observar tanto o comportamento da comunicação quanto a capacidade do sistema de manter sua operação diante de falhas. Neste trabalho, são consideradas métricas de desempenho e confiabilidade, de modo a analisar o funcionamento da arquitetura em condições normais e sob falhas controladas [Avizienis et al. 2004].

A disponibilidade (A) é utilizada para representar a razão entre o tempo em que o sistema permaneceu operacional e o tempo total de execução, sendo expressa em porcentagem. A latência de publicação (L) corresponde ao intervalo entre o envio de uma mensagem MQTT e a confirmação de recebimento pelo broker, permitindo observar o atraso associado à comunicação. O jitter (J) é calculado a partir da variação entre latências consecutivas e permite analisar a regularidade temporal do canal de comunicação [Trivedi and Bobbio 2017].

A taxa de entrega de pacotes (PDR) corresponde à razão entre pacotes confirmados e pacotes enviados, indicando a eficiência da transmissão. O RSSI (Received Signal Strength Indicator), por sua vez, representa a intensidade do sinal WiFi recebido pelo dispositivo, sendo utilizado como indicador das condições de conectividade durante a operação. Além dessas métricas, o tempo de *failover* (T_f) é considerado para medir o intervalo entre a detecção de falha no servidor primário e a conexão ao servidor secundário, enquanto o tempo de recuperação (T_r) expressa o intervalo necessário para o retorno do dispositivo ao servidor primário após o restabelecimento do serviço. Também são observados o MTBF (*Mean Time Between Failures*) e o MTTR (*Mean Time To Repair*), que fornecem apoio à análise da continuidade do serviço e da resposta da arquitetura diante de falhas [Trivedi and Bobbio 2017].

4. Metodologia

Esta seção descreve a metodologia adotada para avaliar o desempenho e a confiabilidade da arquitetura IoT proposta em um ambiente com redundância entre provedores de nuvem. O experimento foi conduzido em um ambiente experimental (*testbed*) composto por um dispositivo embarcado, duas instâncias em nuvem configuradas de forma redundante e serviços de apoio para armazenamento, sincronização, monitoramento e injeção de falhas. A Figura 1 ilustra o fluxo metodológico.

Inicialmente, foi realizada a configuração do ambiente experimental, composto por um microcontrolador ESP32 acoplado a um sensor digital de temperatura e umidade DHT22, além de duas máquinas virtuais hospedadas na Google Cloud Platform (GCP) e na Amazon Web Services (AWS). O DHT22 foi utilizado por fornecer medições ambientais, adequadas para gerar tráfego periódico no experimento com baixo custo e fácil integração ao microcontrolador. Em cada servidor, foi utilizado o Mosquitto como *broker* de mensagens, o InfluxDB para armazenamento de séries temporais e o Grafana para visualização dos dados e métricas. A comunicação entre o dispositivo e os servidores ocorreu por Wi-Fi, com uso do protocolo MQTT.

Na etapa seguinte, foi realizada a coleta e publicação dos dados. Durante a execução, o ESP32 efetuou leituras contínuas e publicou as mensagens no servidor ativo, utilizando um mecanismo de *failover* automático para alternar entre os provedores em caso de falhas consecutivas. Em paralelo, as mensagens foram persistidas localmente em cada servidor e submetidas a um mecanismo periódico de sincronização entre réplicas, com o objetivo de reduzir lacunas de dados após interrupções ou comutações.

A execução dos cenários experimentais foi dividida em três fases: operação normal, operação com falhas injetadas e operação após a desativação da injeção de falhas. Essas fases permitiram comparar o comportamento da arquitetura em diferentes condições de operação. Durante a fase com falhas, o *broker* MQTT do servidor primário foi submetido a interrupções controladas, alternando períodos de funcionamento e indisponibilidade com tempos gerados a partir de distribuição exponencial.

Na etapa de monitoramento e coleta, foram registrados indicadores de desempenho e confiabilidade, incluindo latência, jitter, disponibilidade, taxa de entrega de pacotes, tempo de *failover* e distribuição do tráfego entre servidores. Além dos registros produzidos pelo dispositivo embarcado, foi utilizado um monitor externo para verificar periodicamente a acessibilidade dos *brokers* MQTT e registrar eventos de falha e recuperação. Na etapa de análise, os dados foram organizados em janelas temporais de 10 minutos, intervalo adotado por conciliar resolução temporal e estabilidade estatística nas comparações entre as fases. A partir dessas janelas, foram calculadas estatísticas descritivas, intervalos de confiança de 95% e testes não paramétricos (Kruskal–Wallis e Mann–Whitney) ao nível de significância de 5%. O monitoramento das condições de conectividade sem fio (RSSI) e das variáveis ambientais ao longo de toda a execução permitiu verificar a estabilidade dessas condições durante as três fases. Os scripts utilizados serão disponibilizados em repositório público.

5. Arquitetura Proposta

A arquitetura proposta é composta por um nó de sensoriamento e por duas instâncias em nuvem, organizadas para permitir a continuidade da comunicação e a replicação de dados

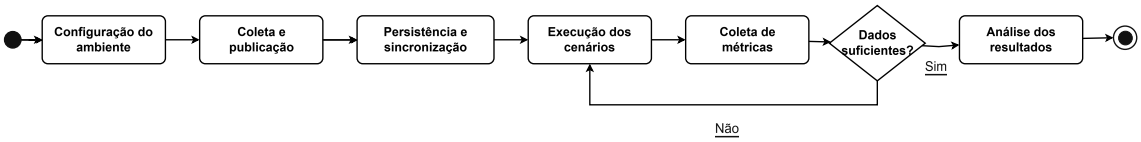

Figura 1. Fluxo metodológico do experimento.

entre provedores distintos. A Figura 2 apresenta a visão geral do sistema. No lado do sensoriamento, um ESP32 acoplado a um sensor DHT22 realiza a coleta de dados e envia as medições por WiFi. No lado da nuvem, a infraestrutura é distribuída entre a Google Cloud Platform (GCP) e a Amazon Web Services (AWS), com cada instância executando Mosquitto, InfluxDB, Grafana e um serviço de bridge.

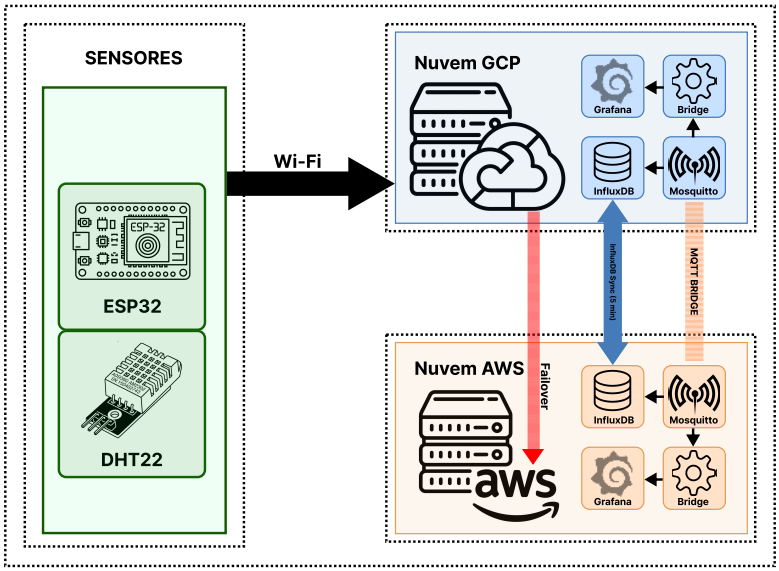

Figura 2. Arquitetura do sistema multi-cloud com *failover* e replicação.

No dispositivo de borda, o firmware realiza leituras periódicas do DHT22 e publica os dados por MQTT. Além das variáveis ambientais, o payload inclui informações de conectividade e operação, permitindo acompanhar o comportamento do sistema durante a execução. A comunicação ocorre por WiFi, com o ESP32 mantendo uma lista ordenada de servidores, em que o GCP é tratado como servidor primário e a AWS como servidor secundário. Quando a publicação ou a conexão falha de forma consecutiva, o dispositivo passa a utilizar o próximo servidor disponível da lista. Após operar no servidor secundário, o firmware verifica periodicamente a possibilidade de retorno ao primário. Na implementação adotada, a comutação entre servidores ocorre quando são observadas três falhas consecutivas de publicação ou conexão MQTT, valor definido como limiar para reduzir trocas excessivas por falhas transitórias e, ao mesmo tempo, permitir resposta rápida à indisponibilidade do servidor ativo. O Algoritmo 1 descreve esse fluxo.

Em operação, o algoritmo executa ciclos sucessivos de leitura, formatação e publicação dos dados. Quando ocorrem falhas consecutivas acima do limiar definido, o firmware realiza a comutação para o próximo servidor da lista e, ao operar no servidor secundário, verifica periodicamente a recuperação do servidor primário.

Algoritmo 1: Publicação MQTT com *failover* automático

Entrada: Servidores $S = \{GCP, AWS\}$, limiar de falhas $k = 3$
Saída: Dados publicados no broker ativo

$i \leftarrow 0$;
$falhas \leftarrow 0$;
enquanto *verdadeiro* **faça**
  $dados \leftarrow$ LerSensores();
  $payload \leftarrow$ FormatarPayload($dados$);
  se *MQTT.Publicar(*$S[i], payload$*)* **então**
    $falhas \leftarrow 0$;
  senão
    $falhas \leftarrow falhas + 1$;
    se $falhas \geq k$ **então**
      $i \leftarrow (i + 1) \bmod |S|$;
      $falhas \leftarrow 0$;
    fim
  fim
  se $i \neq 0$ ***e*** *TempoDeVerificacao()* **então**
    se *TestarTCP(*$S[0]$*)* **então**
      $i \leftarrow 0$;
    fim
  fim
fim

Nas instâncias em nuvem, os dados recebidos pelo broker Mosquitto são encaminhados ao InfluxDB, enquanto o Grafana consulta o banco para visualização em tempo real. Além do fluxo principal de publicação, a arquitetura inclui dois mecanismos complementares de sincronização entre os provedores. O primeiro consiste em uma bridge MQTT entre os brokers, permitindo o encaminhamento de mensagens entre GCP e AWS. O segundo corresponde à sincronização periódica entre os bancos InfluxDB, utilizada para reduzir lacunas de dados após interrupções ou comutações entre servidores. Assim, quando o dispositivo passa a publicar no servidor secundário, a arquitetura continua armazenando os dados e posteriormente busca restabelecer sua consistência entre as duas instâncias. A utilização de dois provedores distintos (GCP e AWS) reduz a dependência de uma única infraestrutura, de modo que falhas em um provedor não comprometam a operação do sistema como um todo.

Para a avaliação da arquitetura em condições de instabilidade controlada, foi incorporado ao ambiente um mecanismo de injeção de falhas que atua diretamente sobre o broker Mosquitto do servidor GCP. Esse componente alterna períodos de operação e de interrupção do serviço, simulando indisponibilidades temporárias no servidor primário e permitindo observar a resposta do sistema quanto à continuidade da publicação, ao redirecionamento do tráfego para a AWS e ao retorno posterior ao GCP. Na implementação adotada, a duração total do experimento é representada por T_{max}, enquanto os intervalos de operação normal e de indisponibilidade são gerados, respectivamente, a partir de distribuições exponenciais com parâmetros médios $\mu_{TTF} = 7$ min e $\mu_{TTR} = 20$ min, valores escolhidos para produzir ciclos de falha e recuperação com frequência e duração suficientes para exercitar o mecanismo de *failover* durante o período experimental. O Algoritmo 2 descreve esse fluxo de injeção de falhas.

Em operação, o ESP32 publica dados no servidor ativo via MQTT, enquanto a bridge e a sincronização entre os bancos mantêm os dados disponíveis nas duas instâncias. Quando o servidor primário se torna indisponível, o *failover* transfere a publicação para o servidor secundário, com retorno ao GCP após sua recuperação.

Algoritmo 2: Injeção de falhas no broker MQTT

Entrada: T_{max}: duração do experimento, μ_{TTF}, μ_{TTR}
Saída: Sequência de eventos de falha e reparo

```
t ← 0;
enquanto t < T_max faça
    TTF ← Exp(μ_TTF);
    Aguardar(TTF);
    PararMosquitto();
    TTR ← Exp(μ_TTR);
    Aguardar(TTR);
    IniciarMosquitto();
    t ← t + TTF + TTR;
fim
```

6. Estudos de Caso

Esta seção apresenta quatro estudos de caso utilizados para analisar experimentalmente o comportamento da arquitetura proposta ao longo das diferentes fases do experimento. O primeiro estudo examina o comportamento da comunicação, com foco em métricas de rede. O segundo analisa o comportamento dos servidores, as condições de conectividade e as variáveis ambientais registradas durante a execução. O terceiro investiga as falhas observadas na fase com injeção de falhas, considerando sua duração e seus efeitos sobre a operação do sistema. Por fim, o quarto compara a arquitetura multi-cloud com o cenário de operação baseado em servidor único ao longo das diferentes fases do experimento.

6.1. Estudo de Caso I

O objetivo deste estudo de caso é analisar o comportamento da comunicação durante o experimento, com ênfase nos efeitos da injeção de falhas sobre a disponibilidade, a latência, a taxa de entrega de pacotes e a distribuição do tráfego entre os servidores. Para isso, as métricas foram agregadas em janelas de 10 minutos, permitindo comparar as três fases do experimento: Fase 1 (operação normal), Fase 2 (operação com falhas injetadas) e Fase 3 (operação pós-falhas), por meio de estatísticas descritivas, intervalos de confiança de 95% e testes não paramétricos. A Figura 3 apresenta as métricas de rede obtidas.

A Figura 3(a) apresenta a disponibilidade em janelas de 10 minutos, com o objetivo de contrastar o comportamento da arquitetura com redundância entre provedores e o comportamento esperado caso a publicação dependesse apenas do servidor primário. Na arquitetura multi-cloud, a disponibilidade média observada foi de 99,28% na Fase 1, 99,65% na Fase 2 e 99,86% na Fase 3, indicando manutenção da continuidade do serviço mesmo durante a fase com falhas injetadas. Já no cenário sem redundância, representado pela dependência exclusiva do GCP, a disponibilidade caiu para 26,37% durante a Fase 2, refletindo o impacto direto das interrupções no servidor primário. A Figura 3(b) apresenta a latência média de publicação MQTT nas três fases do experimento. Na Fase 1, correspondente ao período inicial de operação normal, a média observada foi de 74,8 ms. Na Fase 2, durante a injeção de falhas, esse valor passou para 32,0 ms. Na Fase 3, após a recuperação do sistema, a latência média ficou em 43,3 ms. A redução da latência nas Fases 2 e 3 em relação à Fase 1 está associada à redistribuição do tráfego para a AWS, cujo tempo de resposta foi inferior ao do GCP no período observado, conforme indicado pela distribuição de tráfego na Figura 3(d).

A Figura 3(c) descreve a taxa de entrega de pacotes ao longo do experimento. Nas janelas analisadas, o PDR permaneceu igual a 1,0 nas três fases, indicando manutenção

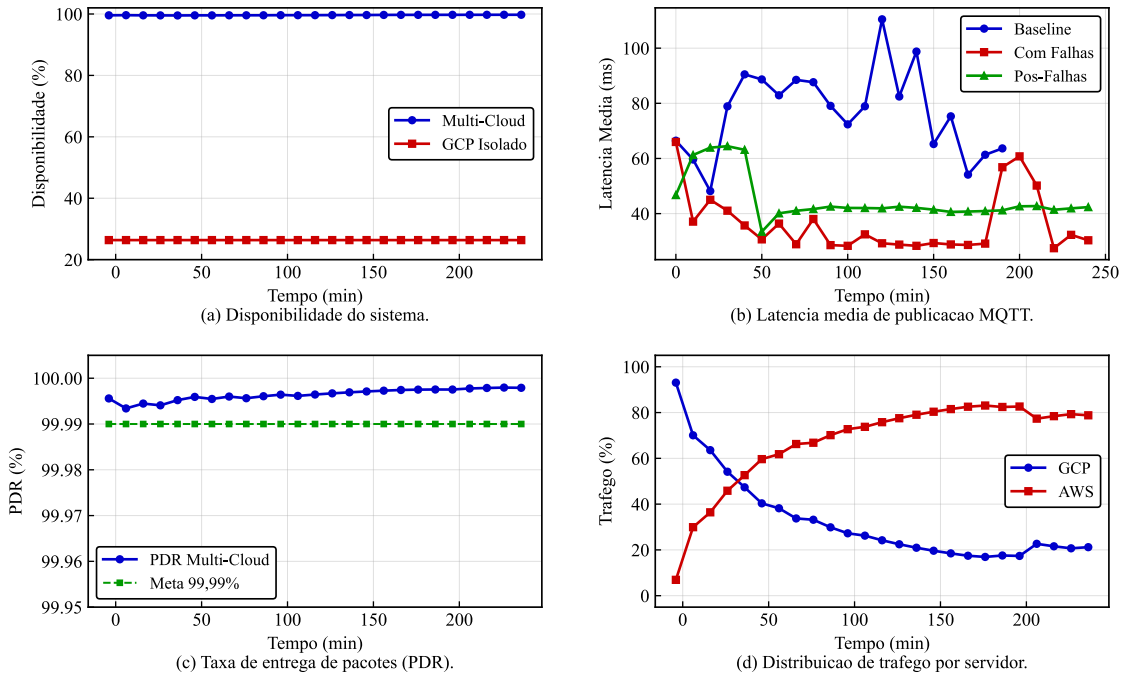

Figura 3. Métricas de rede obtidas ao longo do experimento. Em (a), a disponibilidade da arquitetura multi-cloud é contrastada com um cenário de referência baseado na dependência exclusiva do servidor primário.

da entrega integral das mensagens observadas na base utilizada na análise. A Figura 3(d) mostra como o tráfego foi redistribuído entre GCP e AWS. Durante a fase com falhas, houve redução da participação do GCP e aumento correspondente da AWS, caracterizando a transferência do tráfego para o servidor secundário quando o primário se tornou indisponível. Além da análise descritiva, foram aplicados testes não paramétricos entre as fases do experimento. O teste de Kruskal–Wallis indicou diferenças estatisticamente significativas ($p < 0{,}05$) entre as três fases para latência, jitter, disponibilidade e fração de tráfego encaminhada ao GCP. Comparações pareadas com o teste de Mann–Whitney confirmaram diferenças significativas ($p < 0{,}05$) entre as Fases 1 e 2 e entre as Fases 2 e 3 para essas métricas, com a Fase 2 apresentando 24 janelas de 10 minutos. O PDR não apresentou variação entre as fases por permanecer igual a $1{,}0$ em todas as janelas observadas, não sendo submetido aos testes.

6.2. Estudo de Caso II

O objetivo deste estudo de caso é examinar o comportamento operacional dos servidores durante a fase com injeção de falhas, bem como as condições de conectividade sem fio e as variáveis ambientais registradas ao longo do experimento. A Figura 4 apresenta essas informações.

A Figura 4(a) mostra o estado dos servidores GCP e AWS com base nos registros do monitor de eventos. Durante a Fase 2, o GCP alternou entre períodos de operação e falha, conforme definido no cenário experimental, enquanto a AWS permaneceu disponível na maior parte do tempo, com um intervalo curto de falhas breves. A Figura 4(b) apresenta o jitter médio e o percentil 95 calculados em janelas de 10 minutos durante a fase com falhas. Neste estudo, o jitter é utilizado como indicador da estabilidade temporal

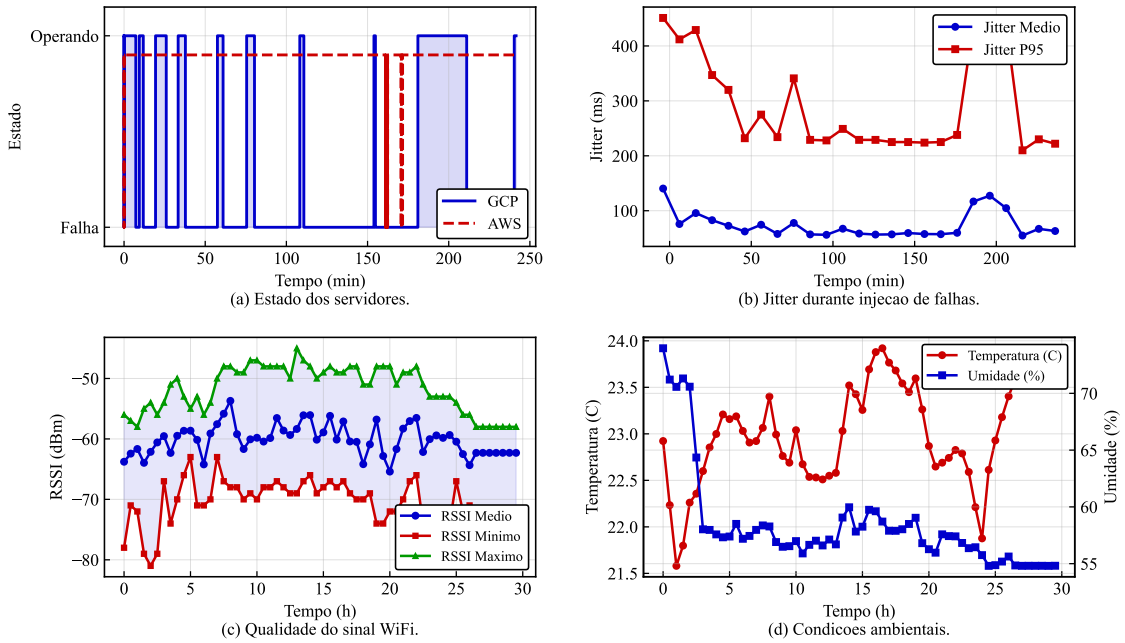

Figura 4. Estado operacional dos servidores, estabilidade temporal da comunicação, conectividade sem fio e condições ambientais.

da comunicação, e não como métrica de confiabilidade. As Figuras 4(c) e 4(d) mostram, respectivamente, a qualidade do sinal WiFi e os valores de temperatura e umidade medidos pelo DHT22. Em conjunto, essas variáveis sugerem que as variações observadas nas métricas de comunicação estiveram mais relacionadas às mudanças de infraestrutura e aos eventos de falha do que às condições ambientais. A estabilidade do RSSI e das variáveis ambientais ao longo das três fases reforça que o comportamento observado nas métricas de rede decorre das interrupções no servidor primário e da consequente redistribuição do tráfego, e não de fatores externos ao experimento.

6.3. Estudo de Caso III

O objetivo deste estudo de caso é caracterizar as falhas registradas durante a Fase 2, considerando sua duração, a disponibilidade acumulada dos servidores e os efeitos observados sobre a operação do sistema. Nesta subseção, a caracterização das falhas é baseada nos registros do monitor de eventos externo, utilizado para identificar os intervalos reais de falha e recuperação dos servidores ao longo da execução. A Figura 5 apresenta a duração das falhas registradas nesse intervalo.

No GCP, foram registrados 9 ciclos de falha, em que cada ciclo corresponde ao intervalo compreendido entre a detecção da indisponibilidade do servidor primário pelo monitor externo e a identificação do restabelecimento do serviço. As durações variaram entre 2,2 minutos, no ciclo F1, e 43,2 minutos, no ciclo F7, resultando em média de 19,8 minutos. Essa variação é compatível com a estratégia adotada para a geração dos períodos de indisponibilidade e reparo. Como os valores medidos pelo monitor incluem o instante de interrupção do serviço, a detecção da falha e a identificação do restabelecimento, podem ocorrer diferenças entre os tempos parametrizados no injetor e os tempos efetivamente registrados.

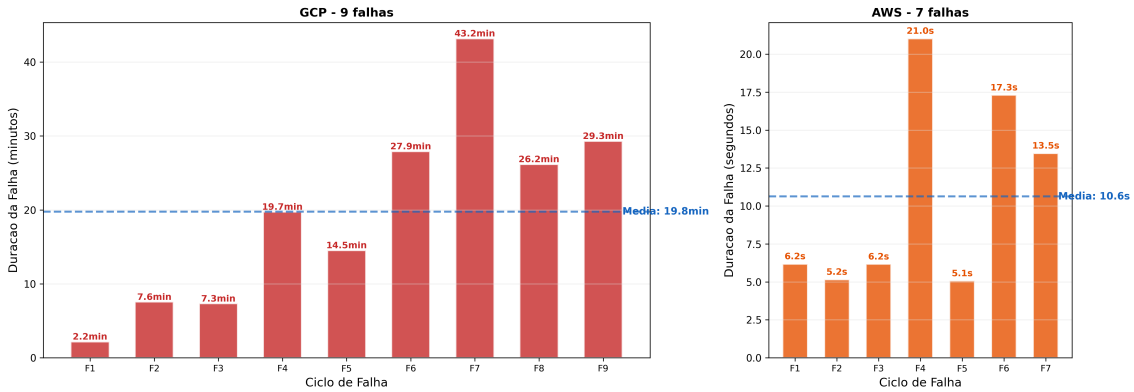

Figura 5. Duração das falhas registradas durante a Fase 2.

Considerando toda a duração da Fase 2, o GCP acumulou aproximadamente 177,8 minutos em estado de falha, o que correspondeu a uma disponibilidade de 26,37% nesse intervalo. Esse resultado evidencia o impacto das interrupções no servidor primário e ajuda a explicar a migração da publicação para o servidor secundário, observada ao longo da fase com falhas. Nesse intervalo, o ESP32 passou a utilizar com maior frequência a AWS, totalizando 17 eventos de *failover*. Com base nos ciclos observados, o MTBF do GCP na Fase 2 foi de aproximadamente 7,1 minutos e o MTTR de 19,8 minutos, valores coerentes com os parâmetros da distribuição exponencial adotada ($\mu_{TTF} = 7$ min e $\mu_{TTR} = 20$ min).

Na AWS, foram registradas 7 falhas breves concentradas em um intervalo de cerca de 10 minutos. Como o mecanismo de injeção atuava apenas sobre o GCP, essas ocorrências não foram produzidas diretamente pelo injetor. As falhas variaram entre 5,1 s e 21,0 s, com média de 10,6 s, resultando em disponibilidade de 99,49% na Fase 2. Essas ocorrências podem estar associadas a instabilidades transitórias na rede ou no serviço, não relacionadas ao mecanismo de injeção.

A Figura 6 mostra a disponibilidade acumulada dos servidores ao longo da Fase 2. No GCP, a curva descreve reduções sucessivas, estabilizando-se em torno de 26% ao final do período. Na AWS, a curva permaneceu próxima de 100%, pois as falhas foram breves e concentradas em um intervalo curto, com impacto reduzido sobre a disponibilidade acumulada.

A Tabela 2 sintetiza os principais indicadores observados na Fase 2 para os dois servidores, incluindo o número de falhas, o tempo total acumulado em indisponibilidade, a disponibilidade e as durações média, mínima e máxima das falhas. Em conjunto, esses resultados mostram que as falhas no GCP foram menos frequentes em número absoluto, porém significativamente mais longas, enquanto a AWS apresentou poucas interrupções e de curta duração.

6.4. Estudo de Caso IV

Este estudo de caso compara a arquitetura proposta com um cenário de referência sem redundância, em que a publicação depende apenas do servidor primário GCP, ao longo das três fases. A Figura 7 apresenta esse comparativo considerando disponibilidade, latência, distribuição de tráfego e taxa de entrega de pacotes.

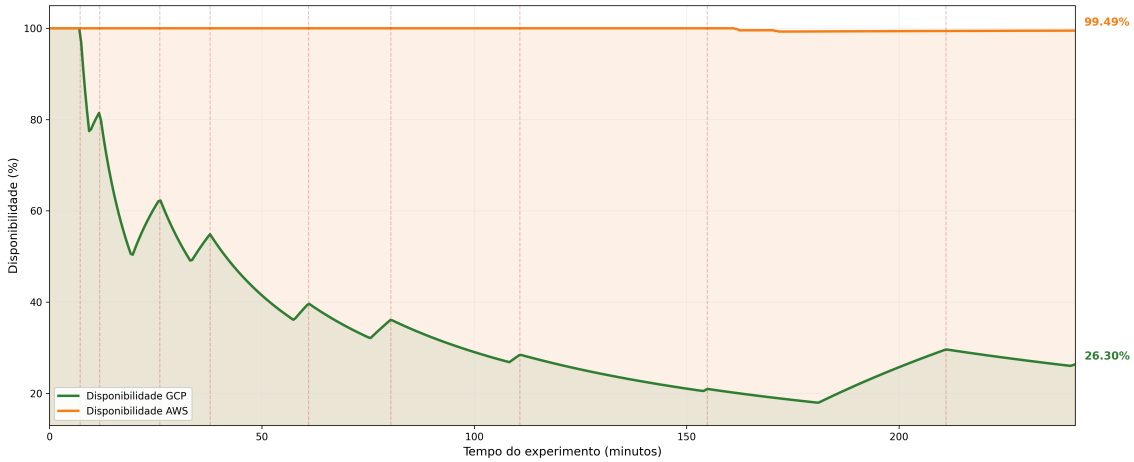

Figura 6. Disponibilidade acumulada dos servidores durante a Fase 2.

Tabela 2. Resumo dos indicadores de falha e indisponibilidade observados na Fase 2.

Métrica	GCP (Primário)	AWS (Backup)
Duração total	242 min	242 min
Falhas registradas	9	7
Tempo em falha	177,8 min	1,2 min
Tempo operando	63,7 min	240,3 min
Disponibilidade (%)	26,37	99,49
Duração média de falha	19,8 min	10,6 s
Maior falha	43,2 min	21,0 s
Menor falha	2,2 min	5,1 s

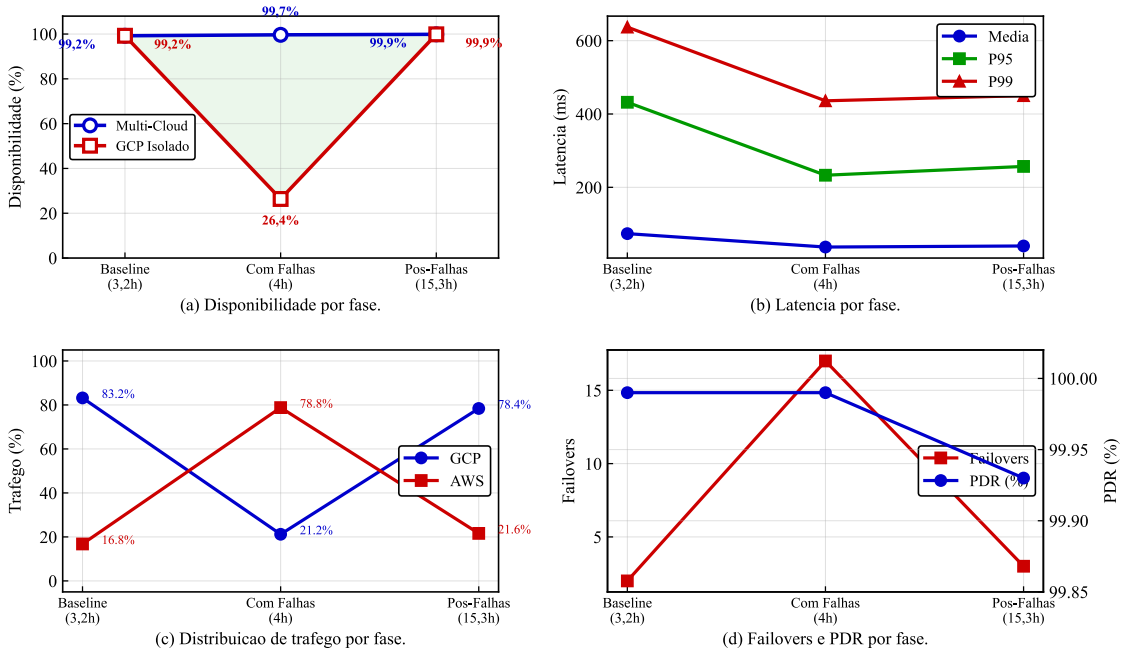

Figura 7. Comparação por fase entre a arquitetura multi-cloud e um cenário de referência sem redundância baseado apenas no servidor primário

A Figura 7(a) mostra a disponibilidade em cada fase, contrastando a arquitetura multi-cloud com um cenário de referência sem redundância baseado apenas no GCP. Na Fase 1, os dois cenários apresentaram valores próximos, pois não houve interrupções relevantes no servidor primário. Na Fase 2, a arquitetura multi-cloud registrou 99,65%, enquanto o cenário sem redundância apresentou 26,37%, evidenciando o impacto da dependência exclusiva do servidor primário. Na Fase 3, com a desativação da injeção de falhas, os valores voltaram a se aproximar.

A Figura 7(b) apresenta a latência média de publicação por fase. Na Fase 1, a latência média observada foi de 74,8 ms. Na Fase 2, esse valor passou para 32,0 ms. Na Fase 3, a latência média ficou em 43,3 ms. A latência menor nas Fases 2 e 3 reflete a maior participação da AWS no tráfego, dado que esse servidor apresentou tempos de resposta mais baixos que o GCP. A Figura 7(c) ilustra a distribuição do tráfego entre os servidores GCP e AWS, evidenciando a migração do tráfego para o servidor secundário durante a fase com falhas e o retorno predominante ao servidor primário na fase pós-falhas.

A Figura 7(d) descreve a taxa de entrega de pacotes em cada fase. Na análise por janelas, o PDR permaneceu igual a 1,0 ao longo das três fases, indicando manutenção da entrega das mensagens observadas.

A comparação mostra que o ganho da arquitetura proposta está na capacidade de preservar a publicação quando o servidor primário se torna indisponível: enquanto o cenário sem redundância sofre degradação na Fase 2, a arquitetura multi-cloud mantém disponibilidade acima de 99% e redistribui o tráfego para o servidor secundário. Na Fase 3, a convergência dos valores entre os dois cenários confirma que a recuperação do servidor primário restabelece o comportamento observado antes da injeção de falhas.

7. Conclusão

Este artigo apresentou uma avaliação experimental do desempenho e da confiabilidade da comunicação em uma arquitetura IoT distribuída voltada ao monitoramento em saúde, baseada em multi-cloud, com *failover* automático e replicação de dados entre provedores de nuvem. A análise mostrou que a redundância entre provedores permitiu preservar a continuidade da publicação e a disponibilidade do serviço mesmo quando o servidor primário foi submetido a falhas controladas.

A análise por janelas temporais de 10 minutos, com testes de Kruskal–Wallis e Mann–Whitney ao nível de 5%, mostrou que a arquitetura multi-cloud manteve disponibilidade acima de 99% em todas as fases (99,28%, 99,65% e 99,86%), enquanto o cenário sem redundância apresentou 26,37% na fase com falhas. A latência média variou entre 32,0 ms e 74,8 ms conforme a fase, com a redução associada à redistribuição do tráfego para a AWS, e o PDR permaneceu igual a 1,0 nas janelas observadas.

Como limitações, o experimento utilizou um único dispositivo e dois servidores, o que limita a generalização dos resultados. A abordagem de injeção atua sobre o serviço (*failure injection*), sem introduzir faltas internas (*fault injection*), e o sensor DHT22 gera dados ambientais, de modo que os resultados refletem o comportamento da infraestrutura de comunicação, não de uma plataforma com sensores de sinais vitais. A avaliação também não abrange escalabilidade, custos ou correlação entre métricas. Como trabalhos futuros, pretende-se ampliar o experimento com múltiplos dispositivos e sensores de

sinais vitais, outras tecnologias de comunicação como LoRaWAN, análise de escalabilidade e custo, diferentes políticas de *failover*, e a correlação entre as métricas avaliadas. Os artefatos experimentais serão disponibilizados em repositório público.

Referências

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33.

Bezerra, T., Callou, G., Airton, F., and Tavares, E. (2025). A modeling-based approach for performance and availability assessment of iomt systems. *Electronics*.

Cao, R., Tang, Z., Liu, C., and Veeravalli, B. (2019). A scalable multicloud storage architecture for cloud-supported medical internet of things. *IEEE Internet of Things Journal*, 7(3):1641–1654.

Lakhan, A., Lateef, A. A. A., Abd Ghani, M. K., Abdulkareem, K. H., Mohammed, M. A., Nedoma, J., Martinek, R., and Garcia-Zapirain, B. (2023). Secure-fault-tolerant efficient industrial internet of healthcare things framework based on digital twin federated fog-cloud networks. *Journal of King Saud University-Computer and Information Sciences*, 35(9):101747.

Nguyen, T., Min, D., Choi, E., and Lee, J.-W. (2021). Dependability and security quantification of an internet of medical things infrastructure based on cloud-fog-edge continuum for healthcare monitoring using hierarchical models. *IEEE Internet of Things Journal*, 8:15704–15748.

Sornlertlamvanich, P., Phakaket, C., Hantula, P., Kanjanawattana, S., Kaoungku, N., and Srivisut, K. (2025). Integration of cloud-based central telemedicine system and iot device networks. *Computers*, 14(9):357.

Stergiou, C. L., Koidou, M. P., and Psannis, K. E. (2023). Iot-based big data secure transmission and management over cloud system: A healthcare digital twin scenario. *Applied Sciences*, 13(16):9165.

Trivedi, K. S. and Bobbio, A. (2017). *Reliability and availability engineering: modeling, analysis, and applications*. Cambridge University Press.

Upadhyay, S., Kumar, M., Upadhyay, A., Verma, S., Kavita, Kaur, M., Khurma, R. A., and Castillo, P. A. (2023). Challenges and limitation analysis of an iot-dependent system for deployment in smart healthcare using communication standards features. *Sensors*, 23(11):5155.

Wadullah Tareq, R. and Ahmed Khaleel, T. (2021). Implementation of mqtt protocol in health care based on iot systems: a study. *International journal of electrical and computer engineering systems*, 12(4):215–223.