ADPS: Providing asynchronous PubSub communication in an adaptive and scalable way in Edge-Fog-Cloud architectures for healthcare data traffic

Guilherme Pohlmann
Unisinos
São Leopoldo, Brasil
gui.pohlmann@hotmail.com

Gabriel Souto Fischer
Unisinos
São Leopoldo, Brasil
gabriel.souto.fischer@gmail.com

Rodrigo da Rosa Righi
Unisinos
São Leopoldo, Brasil
rrrighi@unisinos.br

Cristiano André da Costa
Unisinos
São Leopoldo, Brasil
cac@unisinos.br

Alex Roehrs
Unisinos
São Leopoldo, Brasil
alexr@unisinos.br

ABSTRACT

This document introduces ADPS (Asynchronous Distributed Publish Subscribe), a PubSub-based communication model tailored for a hierarchical architecture distributed on Edge, Fog, and Cloud layers. The primary challenge addressed by this model is establishing runtime connections between subscribers and data providers, enabling communication across various segments. ADPS is implemented within a Smart Cities healthcare application, monitoring vital statistics via IoT devices. Furthermore, this work will cover the specifications of message formats, topic definitions, message delivery reliability and latency effects, broker configurations, control data exchange among nodes, and the dynamic behaviour of topology due to user movement and structural changes with node additions at the Edge and Fog levels. Test results from the model's most demanding scenarios showed latency variations of approximately 200ms, 300ms, and 400ms with quality of service levels 0, 1, and 2 from the initial Fog node to the Cloud.

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; • **Applied computing**;

KEYWORDS

asynchronous communication, internet of things, fog computing, cloud computing, Edge computing

1 INTRODUÇÃO

O ecossistema da Internet das Coisas (IoT) está se tornando mais complexo, com integração entre hardware, software, seres vivos e objetos físicos interagindo para troca e processamento de informações. A IoT está presente em diversos domínios, como cidades inteligentes, casas inteligentes, transporte, fábricas e comércio [10, 16]. Além disso temos a aplicação de IoT na área da saúde, que é chamada de Internet of Medical Things (IoMT) [9] para monitoramento remoto de dados vitais, especialmente relevante no contexto da pandemia de COVID-19, onde o monitoramento é crucial para identificar o avanço do vírus e prevenir situações críticas.

Na engenharia de software, é essencial examinar como as arquiteturas e práticas de design de sistemas estão evoluindo para atender aos novos cenários da IoMT, e avaliar a adequação dos paradigmas e modelos existentes. Uma abordagem promissora é adotar uma arquitetura híbrida que integre Cloud [8, 18], Fog [13, 17] e Edge Computing [2]. No entanto, projetar um modelo de comunicação entre os dispositivos do ecossistema e as diversas camadas, desde os usuários da Edge até os serviços mais complexos na Cloud pode ser um desafio. É crucial entender como essas camadas se comunicam em um sistema hierárquico, mutável e escalável, lidando com a dinâmica de entrada e saída de nós na topologia, a configuração automática de novos nós e a mobilidade geográfica dos usuários.

Diferentes trabalhos já foram publicados no contexto de IoMT para o monitoramento de dados vitais [3, 6, 7, 11, 12, 15, 19, 20], que empregam desde a utilização de arquiteturas com Edge, Fog e/ou Cloud Computing, inteligência artificial, segurança e proteção de dados, até a coleta e gerenciamento de dados vitais dos usuários. Entretanto estes trabalhos não apresentam um modelo aprofundado para a comunicação de tais sistemas, levando em consideração como ocorre a troca de informações em uma solução IoMT que utiliza uma arquitetura hierárquica distribuída na Edge, Fog e Cloud.

A comunicação interna e externa entre diferentes serviços, camadas e servidores é essencial na implementação de uma aplicação IoMT. As lacunas existentes nos trabalhos relacionados oferecem uma oportunidade significativa para pesquisa. Nesse contexto, foi desenvolvido o modelo ADPS para comunicação assíncrona usando o conceito de PubSub para uma aplicação de saúde escalável em Cidades Inteligentes. Diferente de trabalhos relacionados, o modelo estabelece as mensagens a serem trafegadas no sistema, cria a estrutura de tópicos para publishers e subscribers, define os modos de Quality of Service (QoS), configura os brokers de forma distribuída e propõe uma estratégia de orquestração automática para novos nós, considerando a movimentação física do usuário e a escalabilidade do sistema. A principal contribuição científica deste artigo é:

(i) O modelo ADPS adiciona na literatura de Cidades Inteligentes e aplicações de saúde uma arquitetura de comunicação assíncrona, distribuída e hierárquica na Edge, Fog e Cloud, utilizando PubSub.

2 TRABALHOS RELACIONADOS

Muitos estudos foram realizados para tentar resolver problemas relacionados ao monitoramento remoto de sinais vitais dentro de Cidades Inteligentes através de dispositivos inteligentes, no contexto

Tabela 1: Trabalhos relacionados que tratam modelo de comunicação em uma arquitetura hierárquica para o monitoramento de dados vitais e os Critérios avaliados para o modelo MQTT

Artigo	Comunicação	C1 Mensagens	C2 Tópicos	C3 Camadas	C4 Infraestrutura	C5 QoS	C6 Edge	C7 Fog	C8 Cloud
[6]	MQTT	-	-	✓	-	-	✓	-	✓
[7]	MQTT, HTTP	-	✓	-	-	-	✓	-	-
[4]	MQTT	-	-	✓	-	-	✓	✓	✓
[3]	MQTT	-	-	-	-	-	✓	-	✓
[15]	MQTT	-	-	✓	-	-	✓	✓	✓
[19]	MQTT, HTTP	-	-	✓	-	-	✓	✓	✓
[12]	MQTT, HTTP	-	-	-	-	-	✓	-	✓
[11]	MQTT	-	-	✓	-	-	✓	✓	✓
[14]	MQTT	-	-	✓	-	-	✓	✓	✓
[20]	MQTT, HTTP	-	-	-	-	-	✓	-	✓

de comunicação, MQTT (Message Queuing Telemetry Transport) [1] e que utilizam uma arquitetura hierárquica de Edge, Fog e Cloud. O MQTT é um protocolo de comunicação com foco em IoT, utilizado em sistemas amplamente conhecidos como o Facebook Messenger e Whatsapp [5]. Para identificar quais oportunidades estão presentes hoje no contexto desta pesquisa foram considerados os seguintes critérios:

- Formato de mensagens (C1) - São definidos as diferentes mensagens que transitam no sistema, como trafegam, seu conteúdo e formato.
- Definição de tópicos (C2) - É apresentado a modelagem de tópicos MQTT, como é feita a sua criação, seu propósito e quais são seus publishers e subscribers.
- Hierarquia em camadas (C3) - Apresenta diferentes camadas para melhor definir as responsabilidades, redução de latência e conservação de energia.
- Comportamento da infraestrutura (C4) - Trata do funcionamento dos clientes e brokers MQTT em uma arquitetura hierárquica e como eles se conectam de forma automática com mínimo de intervenção manual e adaptação em tempo de execução. Além disso trata da orquestração de novos nós saindo e entrando.
- Modos de qualidade de serviço (C5) - Apresenta os diferentes modos de utilização de QoS do MQTT para o publish e subscribe de mensagens e seu impacto.
- Utiliza Edge (C6) - Considera que os clientes estarão próximos a nós da Edge.
- Utiliza Fog (C7) - Utiliza a Fog como camada para processamento intermediário.
- Utiliza Cloud (C8) - Utiliza a Cloud para pós processamento de dados coletados pelas camadas inferiores e outros serviços mais robustos.

A Tabela 1 lista os trabalhos selecionados que tratam modelo de comunicação em uma arquitetura hierárquica para o monitoramento de dados vitais e os critérios avaliados para o modelo MQTT. Com base na revisão da literatura, fica evidente a importância em entender que tipos de informação transitam nos sistemas, quais os tipos de mensagens que contém dados de usuários, quais contém dados de controle, quais contém dados de configuração e como são as notificações e integração com sistemas externos. Embora existam várias abordagens promissoras na literatura, pode-se notar os modelos observados não apresentam características da troca de mensagens, como seu formato, como transitam no sistema e os diferentes tipos. Além disso os trabalhos não apresentam como utilizam os modos de QoS do MQTT e como seus diferentes níveis impactam o tempo de troca de mensagens, sendo este um detalhe muito importante de implementação pois afeta diretamente o quão confiável é a entrega de mensagens e a latência adicional para se garantir a confiabilidade e evitar perda de informações. Por fim, poucos trabalhos entram em detalhes de como irão funcionar os clientes MQTT e seus brokers, especificamente a questão de funcionamento do sistema de como novos nós entram na rede de forma dinâmica, mobilidade do usuário onde ele se conecta em diferentes nós dependendo da sua localidade e setup automático da aplicação.

3 MODELO ADPS

O ADPS é um modelo de comunicação que permite que dados vitais coletados por dispositivos IoT sejam trafegados pelas diferentes camadas de uma aplicação. O projeto VitalSense [17] é utilizado como caso de uso para embasar as decisões e implementação do modelo proposto. O projeto visa o desenvolvimento de uma aplicação que ofereça serviços de saúde a partir do monitoramento remoto e captura de sinais vitais em tempo real, oferecendo reações quando detectada alguma degradação da saúde de uma pessoa, predição, pós processamento de dados de saúde dentre outras funcionalidades. O projeto está inserido no contexto de Cidades Inteligentes, e utiliza uma arquitetura híbrida distribuída na Edge, Fog, e Cloud de forma hierárquica de forma a dividir responsabilidades e prover melhor qualidade de serviço através da localização e redução de latência. Para atingir os objetivos deste trabalho foram tomadas as seguintes decisões de projeto:

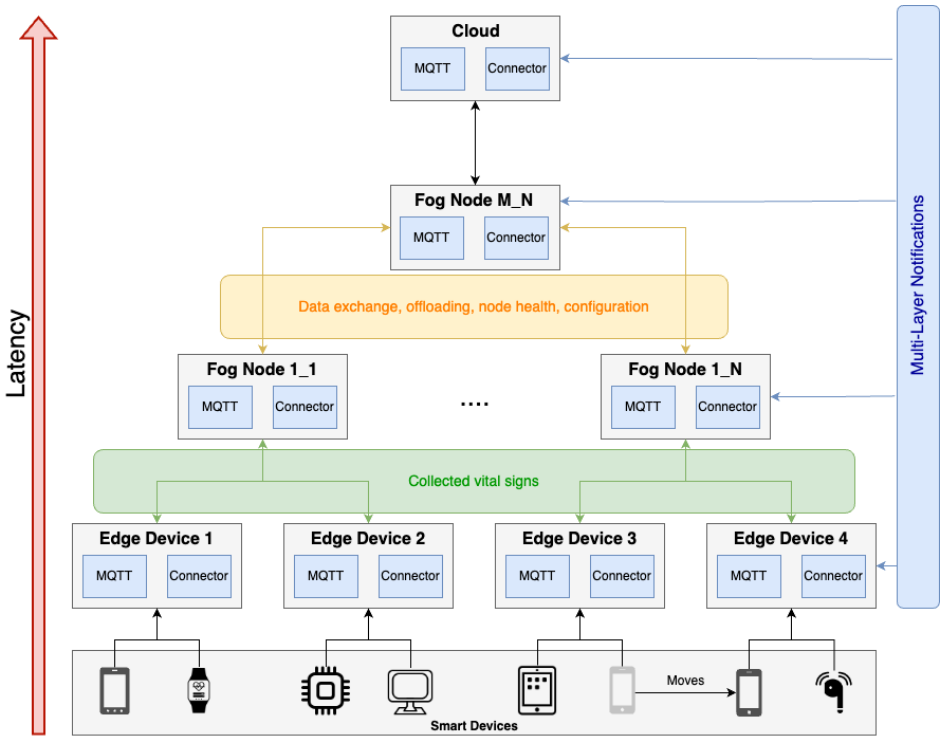

Figura 1: Arquitetura alto nível de comunicação distribuída proposta

- O modelo tem uma arquitetura hierárquica de três camadas (Edge, Fog e Cloud).
- O MQTT será utilizado para comunicação assíncrona não bloqueante entre os nós.
- Os nós e camadas devem possuir hierarquia, de modo que mensagens possam subir e descer no sistema.
- Os nós devem ser capaz de fazer offload para outros nós caso não tenham capacidade computacional, enviando a carga de trabalho para seu nó pai.
- A configuração de novos nós é feita de forma automática. As configurações podem ser mudadas dinamicamente e nós são notificados da mudança.
- Cada nó tem visibilidade de quais são seus nós inferiores e nó superior, para que o fluxo de mensagens bidirecional seja possível.
- A arquitetura deverá lidar com a mobilidade do usuário, alocando o melhor nó para o mesmo se conectar e fazer o publish/subscribe de suas mensagens.

3.1 Arquitetura

A arquitetura de comunicação do modelo ADPS é distribuída em três camadas (Edge, Fog e Cloud). Com isso, o foco deste modelo é a comunicação utilizando MQTT entre os diferentes componentes de cada camada. Além disso, outro requisito muito importante é a QoS em relação ao tempo de resposta na troca de dados, com isso foi incluída a Fog como camada intermediária entre a Edge e a Cloud. Os nós da Fog estarão mais próximos dos usuários que estão na Edge, podendo processar dados vitais e reagir de maneira rápida em casos de anomalias no quadro de saúde enviando notificações e alertas. A Fog também permite a localização e rastreabilidade dos dados e através da comunicação bidirecional a informação é capaz de percorrer diferentes nós.

A camada Edge é onde se encontra os dispositivos como celulares, equipamentos vestíveis, sensores e demais dispositivos IoT. Estes dispositivos irão coletar os dados vitais dos usuários e se conectarão com o nó Fog mais próximo. Toda a orquestração de conexão é automática em função da mobilidade do usuário, caso este se afaste demais do seu Fog Node é necessário estabelecer uma nova conexão com um outro nó mais próximo.

Por fim, a Cloud será a última camada a receber informações das camadas inferiores e estas chegarão de maneira assíncrona a fim de não bloquear os serviços inferiores. O objetivo é depender o menos possível da Cloud para serviços que executam em tempo real, em função da latência. A Cloud irá fornecer pós-processamento para visualização através de dashboards, mapeamento, histórico, dentre outros serviços que não têm criticidade de tempo de execução. A Figura 1 apresenta a arquitetura de alto nível do modelo de comunicação distribuída entre os componentes da Edge, Fog e Cloud, enquanto que a Figura 2 apresenta uma representação visual da arquitetura e serviços implementados.

3.2 Funcionamento

O Serviço de Configuration é responsável por gerenciar as configurações do Connector, sendo elas a de execução da aplicação

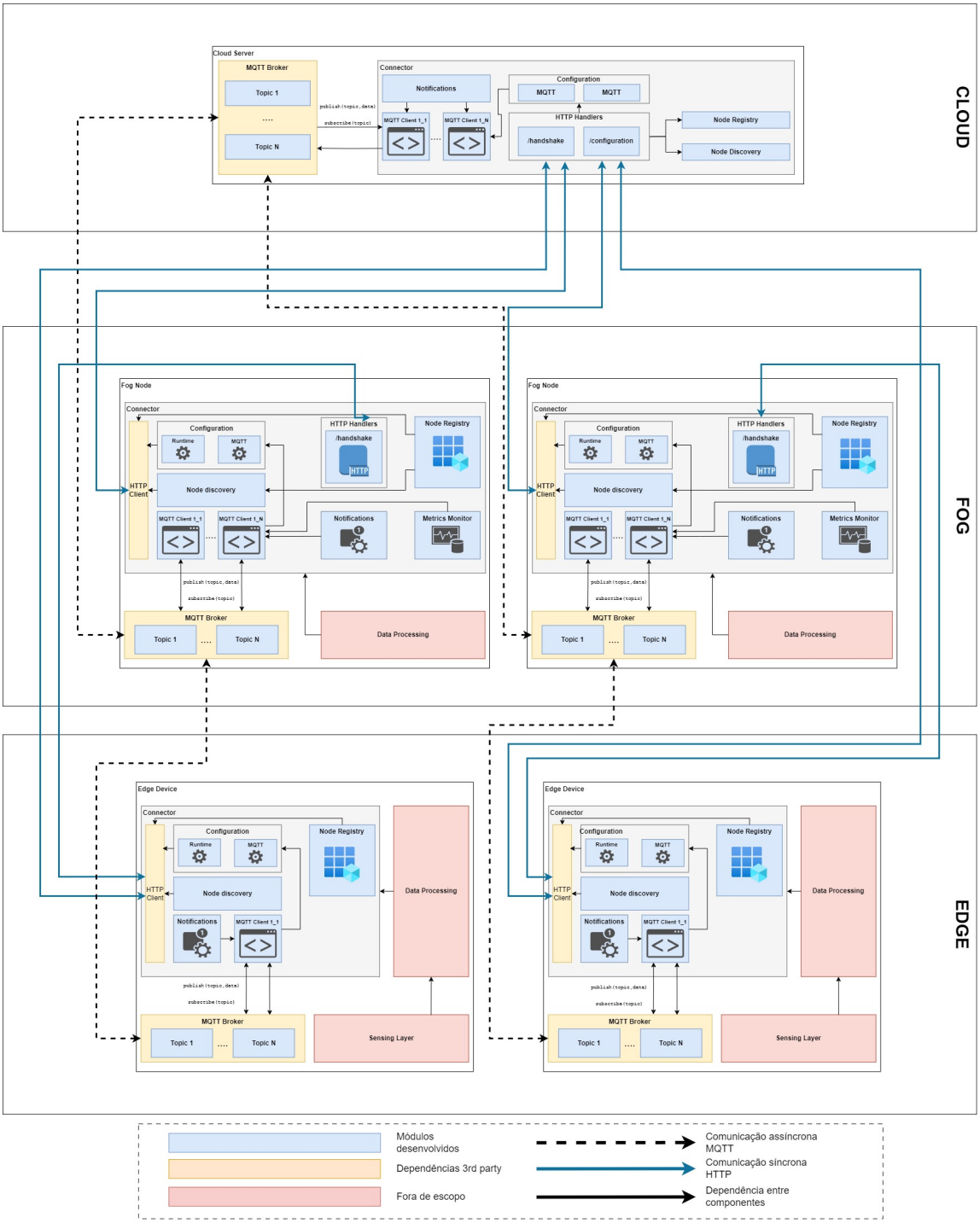

Figura 2: Serviços e arquitetura do modelo

e de clientes e broker MQTT. O serviço tem duas dependências, a primeira é de uma interface HTTP responsável por buscar as configurações de forma síncrona na Cloud para o setup inicial do Connector. A outra dependência é do cliente MQTT, isto porque as configurações podem ser alteradas na Cloud e a mudança será enviado para todos os nós via MQTT para que eles possam atualizar as suas configurações de forma dinâmica. É importante destacar que a dependência do serviço de configuration é da Edge e Fog para

Cloud, o serviço executando na Cloud é a fonte para configuração de runtime e MQTT dos Connectors Fog e Edge.

O Serviço de Node Discovery é responsável por criar a hierarquia entre as camadas, decidindo qual será a conexão entre os nós. Ele tem a responsabilidade de registrar com a Cloud que o nó atual está ativo, pronto para se conectar com seus vizinhos e qual nó mais apropriado para atendê-lo. A Cloud irá registrar esse novo nó e ela possui a visibilidade do todos os nós atuais que estão executando, a partir dessa lista o Node Discovery da Cloud pode decidir qual o melhor nó superior que pode servir esse novo nó que está sendo incluído na hierarquia. Para Edge esse serviço contempla a mobilidade do usuário, visto que a Cloud pode oferecer o melhor nó Fog para se conectar baseado em localização.

O Serviço de Node Registry tem a responsabilidade simples de ter visibilidade de quais nós estão conectados com o nó corrente, isto é importante para permitir o fluxo de mensagens bidirecional para nós superiores e inferiores. Quando o serviço de Node Discovery retorna o nó para se conectar, ele será mantido em memória pelo Node Registry e este também irá criar o cliente MQTT para troca de mensagens entre os dois nós.

O Serviço de Notifications é responsável por publicar mensagens de notificação em caso de anormalidades, caso a camada de processamento de dados vitais identifique uma anormalidade o nó que identificou isso é capaz de gerar um alerta para o usuário mas também subir essa alerta na hierarquia e/ou para algum sistema externo como um hospital. Este serviço depende do cliente MQTT para fazer o publish da notificação.

O Serviço de Metrics Monitor coleta métricas de uso de recursos computacionais do nó atual e compartilha com seus vizinhos através do tópico de Fog health. Este serviço é importante internamente para gerenciar a saúde do sistema e auxiliar com as tarefas de offloading, os nós conhecem a saúde de seus vizinhos e baseado nas configurações de execução sabem como lidar com casos onde offloading é necessário.

Para a configuração das mensagens serão utilizados quatro tópicos. O primeiro serve para receber a publicação de dados vitais coletados pelos dispositivos inteligentes. O segundo tópico é para o processamento de notificações do sistema, que podem ser geradas a partir da análise de sinais vitais. O terceiro tópico é para o recebimento de dados de configuração que podem mudar durante o runtime dos nós. Por fim, o último tópico é utilizado para a troca de dados de controle, útil principalmente para offload de tarefas. Para o trabalho em conjunto de todos os nós de uma camada é extremamente importante que os nós tenham conhecimento da saúde dos seus vizinhos. A estrutura de cada tópico, atributos da mensagem e o que representam estão representados na Figura 3.

Por fim, temos temos um handler muito importante em cada Connector da Edge, Fog e Cloud que é o Handshake. Este endpoint faz a troca de configuração entre os dois nós para que os dois brokers MQTT se conectem e os dois nós possam fazer publish/subcribe entre si e trocar mensagens. Este é um fluxo crítico do sistema que envolve uma requisição HTTP síncrona no início da execução de um Connector, esse setup automático é de grande vantagem para escalabilidade e exclui a necessidade de intervenção humana para a criação de novos nós.

Por se tratar de uma arquitetura hierárquica, os nós da Fog poderão ter seu Fog Connector com múltiplas instâncias de clientes

Vital Signs Topic	
serviceName	Contém o nome do serviço que irá processar aquele sinal vital.
servicePriority	Prioridade do serviço que irá executar aquele sinal vital.
hr	Frequência cardíaca.
rr	Frequência respiratória.
hrv	Variabilidade de frequência cardíaca.
spo2	Saturação de oxigênio.
bt	Temperatura corporal.
userPriority	Prioridade do usuário, um usuário com maior risco pode ter prioridade maior.
userId	Identificador do usuário.
id	Identificador da mensagem.
timestamp	Tempo que foi gerada a mensagem.
metadata	Dados adicionais para melhor processamento, como latitude e longitude do usuário e IP que originou a mensagem.

Notifications Topic	
message	Contém o nome do serviço que irá processar aquele sinal vital.
notificationType	Prioridade do serviço que irá executar aquele sinal vital.
priority	Frequência cardíaca.
userId	Identificador do usuário.
id	Identificador da mensagem.
timestamp	Tempo que foi gerada a mensagem.
metadata	Dados adicionais para melhor processamento.

Node Health Topic	
nodeIp	Endereço IP do nó que enviou suas métricas.
cpu	Uso de CPU atual.
memory	Uso de memória atual.
avgProcessTime	Tempo médio de processamento de dados vitais.
id	Identificador da mensagem.
timestamp	Tempo que foi gerada a mensagem.
metadata	Dados adicionais para melhor processamento.

Configurations Topic	
newParentIp	Endereço IP do novo nó superior para se conectar.
oldParentIp	Endereço IP do antigo nó superior para substituir.
childNodes	Lista de nós inferiores atualizada.
cpuOffloadThreshold	Parâmetro para fazer offload baseado no uso de CPU.
memoryOffloadThreshold	Parâmetro para fazer offload baseado no uso de memória.
id	Identificador da mensagem.
timestamp	Tempo que foi gerada a mensagem.
metadata	Dados adicionais para melhor processamento.

Figura 3: Tópicos MQTT do modelo ADPS

MQTT em memória onde cada um está conectado com o broker MQTT de um nível abaixo da hierarquia. Além disso o Fog Connector possui uma instância de cliente que conecta com o broker MQTT pai, que seria um nó acima na hierarquia ou a Cloud. Este encapsulamento permite que as mensagens permaneçam localizáveis e tenham contexto, isto é importante para o aspecto de mobilidade do usuário e localização dos dados dentro de uma cidade. Esta separação de troca de mensagens baseada em localização é importante para escalabilidade visto que nós da Fog de regiões com menos usuários não irão publicar mensagens em tópicos de regiões que estejam com uma demanda maior.

O primeiro para a inserção de um novo nó na Fog é o handshake entre ele e os nós existentes, esse processo de setup de clientes que irão se comunicar com os brokers será feita de maneira síncrona através de request e response. Os nós da Fog irão depender de serviços de configuração que estão armazenados na Cloud, isto porque os nós da Fog podem depender de um gateway do provedor Cloud que possui um IP estático para acessar as configurações do MQTT, para saber com quais outros nós ele deve se conectar e as demais configurações necessárias para o runtime da aplicação. Os nós da Fog não podem depender uns dos outros durante seu setup devido a mutabilidade destes servidores.

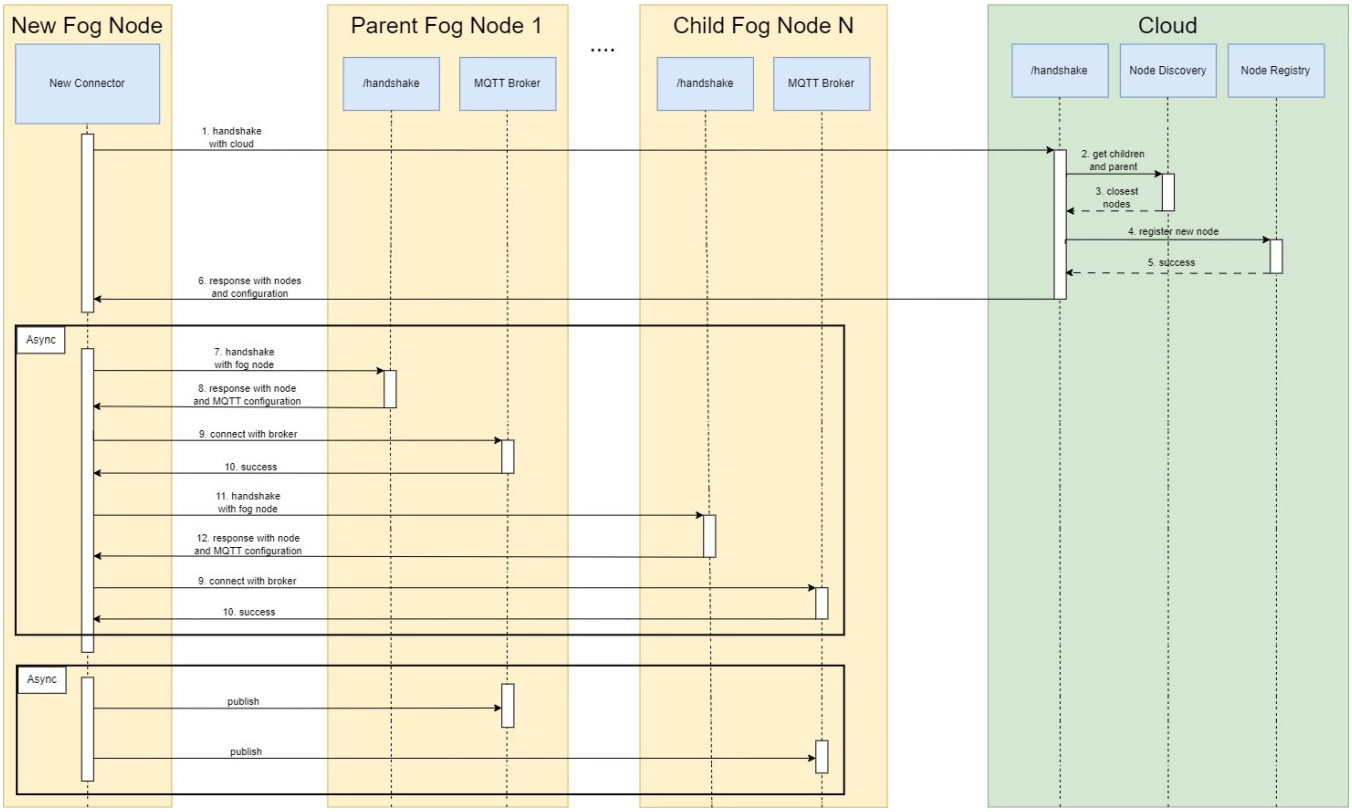

Figura 4: Diagrama de sequência de setup automático de um novo Fog node

Essa lógica também se aplica para nossos dispositivos na Edge, uma vez que a Cloud possui recursos de processamento maior ela pode decidir a qual nó da Fog o usuário irá se conectar. Com base no handshake inicial entre o Edge Connector e os serviços de configuração da Cloud é possível determinar qual nó é o mais apropriado para servir aquele usuário, baseado em localização, latência e balanceamento de carga. O fluxo descrito acima é representado no diagrama de sequência da Figura 4.

Manter a privacidade e a segurança é crucial ao gerenciar as informações de pacientes [21]. Os Fog nodes do VitalSense criam um par de chaves RSA 2048 para criptografia de dados na camada Edge. Os dados são criptografados por Edge controllers usando a chave pública e descriptografados por Fog nodes usando a chave privada. Além disso, o VitalSense utiliza algoritmos de criptografia homomórfica nesta camada para garantir a segurança de todos os dados em trânsito. Os dados brutos podem ser enviados para serviços externos de terceiros e nuvens públicas capazes de realizar operações aritméticas nos dados sem acesso direto. Essas entidades não podem visualizar os dados reais e devem solicitar a descriptografia de seus resultados computacionais de um Fog node.

4 METODOLOGIA DE AVALIAÇÃO

Para execução dos cenários de testes foi implementado um protótipo do modelo ADPS utilizando o broker MQTT Eclipse Mosquitto. A linguagem de programação utilizada para os serviços foi o Golang v1.20. A integração entre os serviços implementados em Golang e o broker Eclipse Mosquitto foi realizada através da biblioteca Paho MQTT. Foi utilizado o JMeter para medir performance e executar testes de carga. Com o intuito de facilitar o deployment dos serviços e broker MQTT foi decidido utilizar Docker containers. Foram criadas duas imagens, uma para o Connector que executa na Edge e Fog e uma para o serviço de Connector da Cloud. A partir dessas duas imagens, os serviços implementados podem ser facilmente executados em um cluster da Cloud ou em qualquer dispositivo que tenha suporte ao Docker. Por fim, os serviços Cloud foram disponibilizados em um cluster da Azure localizado em São Paulo enquanto que os Connectors de Edge e Fog foram executados localmente.

Um dos principais parâmetros utilizados é o modo de QoS do cliente MQTT para a publicação de mensagens, visto que com a maior confiabilidade de entrega de mensagens maior será o overhead de comunicação. Serão testados os diferentes modos de operação e como eles impactam a qualidade de entrega da mensagem, levando em consideração a latência e o seu custo computacional. Além disso serão utilizadas diferentes tipos de mensagens com tamanhos variados, algumas mensagens terão mais informações além dos dados vitais, como por exemplo a localização da origem da mensagem, métricas do nó que enviou as mensagens e outros metadados.

Para a geração de mensagens com sinais vitais e o intervalo entre cada mensagem foi utilizado um conjunto de dados vitais randomizados. Para a validação do modelo o mais importante é a mensagem e sua estrutura, os sinais vitais não precisam ser precisos

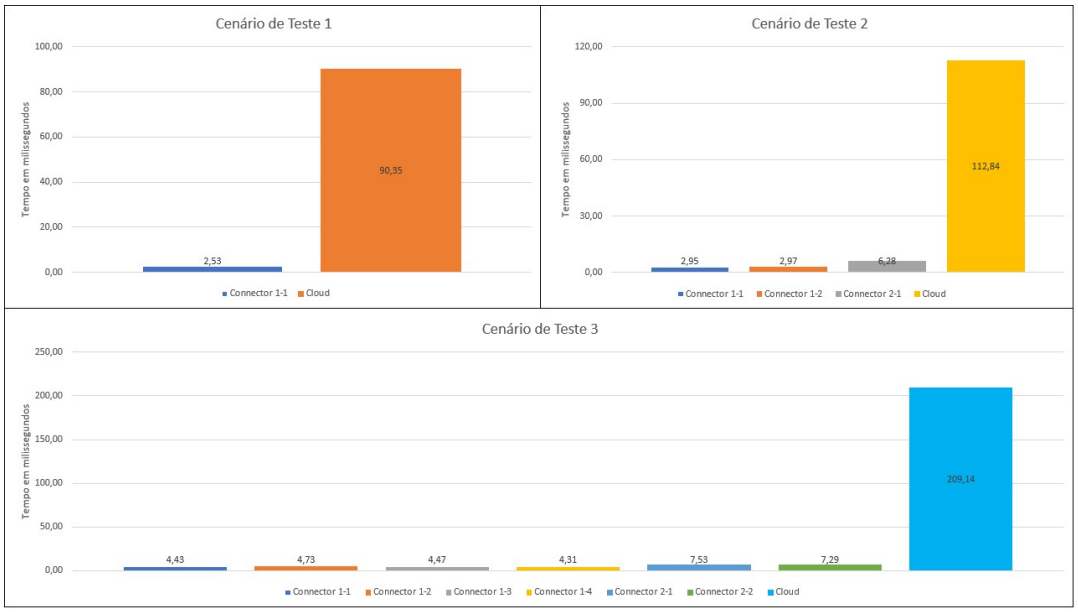

Figura 5: Resultados dos cenários de testes (1) Modelo de uma casa, (2) Modelo de um bairro e (3) Modelo de uma cidade utilizando QoS 0

necessariamente. Além disso, cada dispositivo da Edge irá enviar uma mensagem com uma leitura vital a cada 500 milissegundos. Também serão utilizados diferentes modos de QoS do MQTT de forma a identificar como a confiabilidade de entrega de mensagens impacta na latência de recebimento de mensagens.

Para avaliar o modelo, foram consideradas as seguintes métricas: *(i) Latência*; e *(ii) Tempo de inicialização automática de Connectors*. Para a primeira métrica é esperada uma baixa latência devido a arquitetura e também a modelagem de mensagens. Já para a segunda métrica espera-se avaliar o tempo de setup de um novo Connector para que o mesmo esteja pronto para receber, processar e encaminhar mensagens. Para avaliar o modelo proposto, foram considerados quatro cenários distintos:

- **Cenário 1: Modelo de uma casa** com dois dispositivos vestíveis conectados a um Edge Controller para receber dados vitais de dois usuários, que esta conectado a um Fog Connector, com dados vitais sendo enviados de forma assíncrona ao Cloud Connector para pós-processamento.
- **Cenário 2: Modelo de um bairro** com quatro dispositivos vestíveis enviado mensagens para dois Edge Controllers. Neste caso cada Edge Controller irá servir dois usuários de forma a balancear a carga. Na camada de Fog temos dois Connectors próximos a Edge onde temos uma relação um-para-um entre Edge Controller e Fog Connector. Além disso, existe um Fog Connector superior aos dois próximos a Edge que receberá as mensagens de seus filhos para fazer o encaminhamento para a Cloud.
- **Cenário 3: Modelo de uma cidade** com oito dispositivos vestíveis, quatro Edge Controllers sendo servidos por quatro Fog Connectors em uma relação um-para-um e dois Fog Connectors que servirão os quatro Fog Nodes inferiores e irão encaminhar as mensagens vindo dos nós inferiores para a Cloud.
- **Cenário 4: Inicialização automática de um novo nó** focando em avaliar o tempo de setup de um novo Connector na Fog. Neste cenário foram instanciados 50 Connectors para avaliar o tempo de interação com a Cloud, criação de variáveis para execução de serviço e interação com o broker MQTT local. O intuito deste cenário é demonstrar a capacidade do modelo de escalar rapidamente, podendo instanciar novos nós e inseri-los na hierarquia para servir os usuários.

5 AVALIAÇÃO E DISCUSSÃO DOS RESULTADOS

A Figura 5 apresenta os resultados dos cenários 1, 2 e 3 executando o modo de QoS 0 do MQTT onde é feito o melhor esforço para entrega da mensagem onde o *publisher* não espera uma resposta de que sua mensagem foi recebida. As médias da latência de cada nó demonstram que o tempo de uma mensagem que vai da Edge a Fog é extremamente baixo e o principal gargalo na comunicação está nas mensagens que chegam até o servidor Cloud. Vale a pena notar também que à medida que a infraestrutura cresce e mais nós são adicionados na hierarquia o tempo de tráfego da mensagem aumenta até chegar nas camadas superiores. Além disto, o maior número de mensagens e nós em cada camada contribuiu com a exaustão de recursos computacionais, isto consequentemente favoreceu a maior latência para a transmissão das mensagens para a rede.

Os resultados da Figura 6 apresentam a execução dos cenários 1, 2 e 3 executando o modo de QoS 1 do MQTT onde a entrega da mensagem é garantida pelo menos uma vez. Desde o cenário 1 que possui o menor tráfego por conta do menor número de nós já

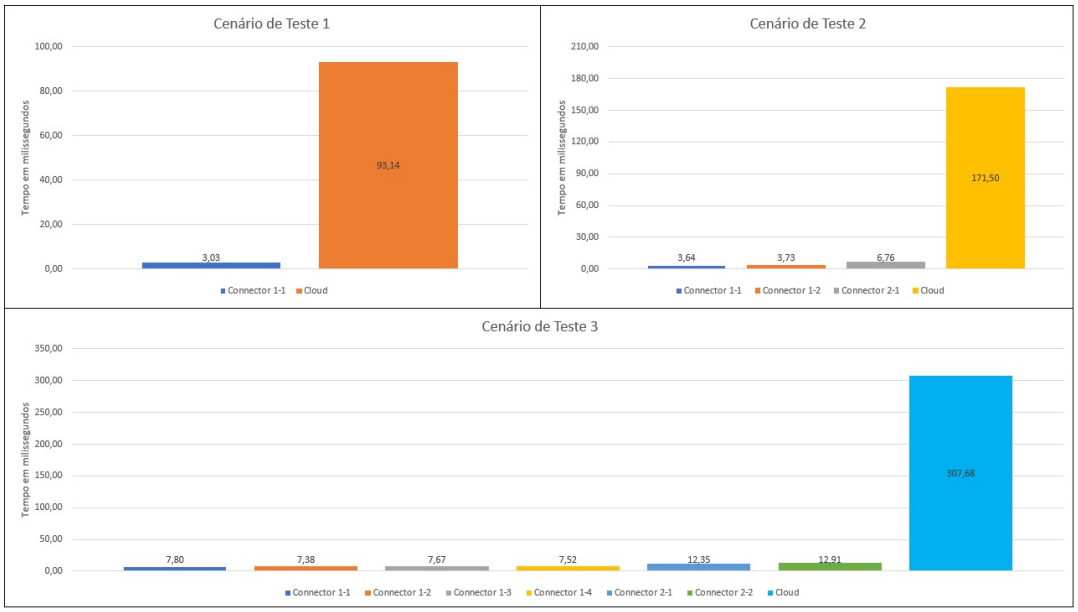

Figura 6: Resultados dos cenários de testes (1) Modelo de uma casa, (2) Modelo de um bairro e (3) Modelo de uma cidade utilizando QoS 1

é evidente a latência adicional por conta da requisição adicional de reconhecimento para a garantia de entrega da mensagem. Conforme a infraestrutura e a demanda por recursos computacionais cresce é possível observar a degradação do tempo de resposta. Além do tempo de transmissão de envio da mensagem existe o tempo de espera para o recebimento do ACK da mensagem e possíveis duplicações desnecessárias de retransmissão quando se opera no nível 1 de QoS do MQTT.

A Figura 7 apresenta a execução dos cenários 1, 2 e 3 executando o modo de QoS 2 do MQTT onde a entrega da mensagem é garantida uma vez e não existe duplicação de mensagens. Neste modo de operação foi onde se observou o maior aumento de latência para o tráfego das mensagens. Este cenário demonstra a análise que deve ser feita quando se escolhe optar por menor tempo de resposta em contra partida da confiabilidade do serviço e vice-e-versa. O nível 2 de QoS possui um handshake de quatro requisições com o broker MQTT, isto fica evidente no tempo para uma mensagem chegar no nível 1 da Fog (Connector 1-N), até o nível 2 (Connector 2-N) e finalmente ao servidor da Cloud onde temos a maior latência.

Por fim, a Figura 8 apresenta a execução dos testes para avaliar o tempo de setup de um novo Connector na Fog. Todos os 50 Fog Nodes que foram inseridos na hierarquia tiveram um startup abaixo de 500ms, demonstrando que o ADPS é escalável e pode se adaptar rapidamente a um aumento de demanda. A maior dependência identificada durante o início de um novo Connector é com a Cloud devido a maior distância em relação ao Fog Node. O handshake entre Fog Nodes é menor, em função da proximidade geográfica.

Com base nos resultados, é possível ver que o modelo é altamente escalável, com tempo para inserção de um novo nó abaixo de 500ms, permitindo que o sistema atenda e reaja a um aumento de demanda repentino. Este é um ponto que pode suprir a limitação computacional de cada Fog Node, visto que conforme o tráfego de mensagem em um nó da Fog aumenta há uma degradação na latência. Isso pode ser um cenário reproduzível no mundo real, onde diversos usuários podem estar centralizados em uma mesma área e há poucos nós para atender todas as requisições, sendo necessário adicionar mais nós para o balanceamento de carga. Ressaltamos que para o contexto proposto não é possível realizar uma análise quantitativa em comparação com os trabalhos relacionados, uma vez que os ambientes e contextos de cada trabalho são diferentes. Portanto, propomos uma análise qualitativa comparando o modelo ADPS com trabalhos relacionados a fim de apresentar as características que nosso modelo propõe que o diferenciam de outros trabalhos da literatura a fim de destacar a contribuição do ADPS para a área. Dessa forma, a Tabela 2 apresenta uma continuação para Tabela 1 de forma a comparar qualitativamente os Trabalhos Relacionados com o modelo ADPS proposto.

Tabela 2: Continuação da Tabela 1, comparando qualitativamente o modelo ADPS proposto com os Trabalhos Relacionados

Artigo	Comunicação	C1	C2	C3	C4	C5	C6	C7	C8
ADPS	MQTT, HTTP	✓	✓	✓	✓	✓	✓	✓	✓

6 CONCLUSÃO

Este artigo apresentou o ADPS, um modelo que contempla a mutabilidade das camadas de Edge e Fog e a mobilidade do usuário para uma aplicação de monitoramento remoto de dados vitais em Cidades Inteligentes, através de uma arquitetura de comunicação

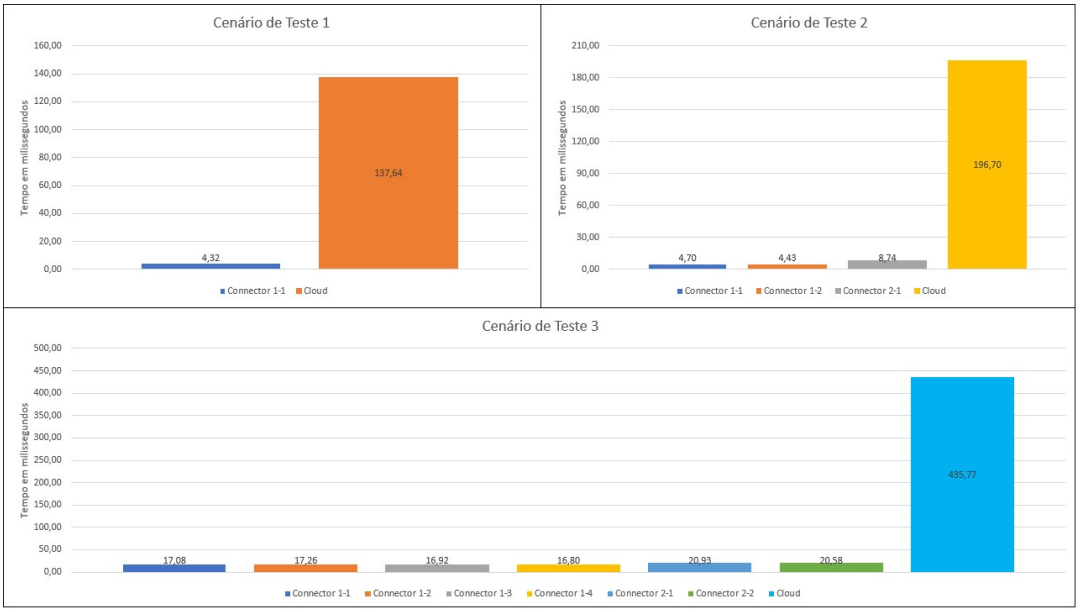

Figura 7: Resultados dos cenários de testes (1) Modelo de uma casa, (2) Modelo de um bairro e (3) Modelo de uma cidade utilizando QoS 2

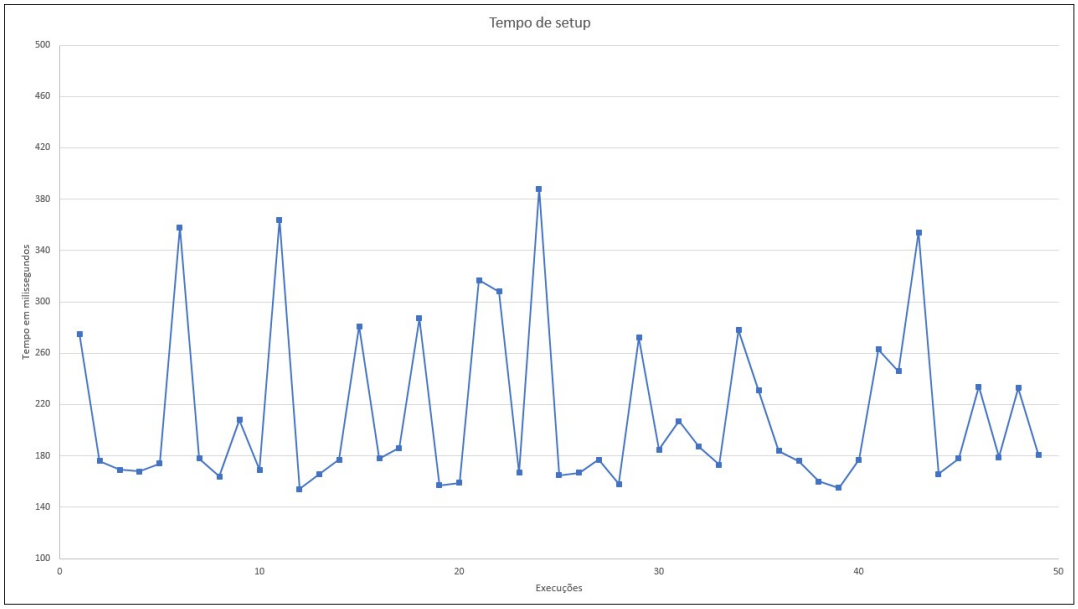

Figura 8: Resultados do cenário de teste 4 do modelo ADPS

utilizando MQTT. Apesar do modelo ter como foco a área da saúde, ele é altamente reutilizável em sistemas com outros propósitos em Cidades Inteligentes, tendo apenas como esforço a mudança do processamento das mensagens de acordo com a necessidade do sistema. Dessa forma, o modelo ADPS traz como principal contribuição científica uma arquitetura de comunicação assincrônica, distribuída e hierárquica na Edge, Fog e Cloud, utilizando PubSub. Ao contrário dos trabalhos relacionados, o modelo ADPS apresenta as mensagens que serão trafegadas no sistema e seus formatos, os tópicos criados dinamicamente, evidencia como os diferentes modos de QoS do MQTT impactaram a performance do modelo, a configuração distribuída dos brokers MQTT e, por fim, a adaptabilidade automática para novos nós que entram na topologia considerando a movimentação física do usuário e escalabilidade do sistema. Com base nos

resultados atingidos o modelo atingiu os objetivos esperados. Foi possível evidenciar as vantagens da Fog para redução no tempo de resposta que é crítico para aplicações de saúde em tempo real, tendo a maior latência identificada para o pior cenário abaixo de 20 milissegundos.

Atualmente o modelo está sendo implementado para testes na cidade São Leopoldo no âmbito do projeto FAPERGS MinhaHistóriaDigital. A título de trabalhos futuros, vislumbra-se o *offloading* de tarefas entre os nós da Fog, balanceamento de carga entre Edge e Fog, filtragem e agregação de dados na Fog para redução de mensagens trafegadas e, por fim, o uso de técnicas de programação paralela para processamento das mensagens pelos *subscribers*. Espera-se também testar os cenários deste trabalho em proporções reais, como por exemplo, diversos nós da Fog distribuídos geograficamente e usuários em diferentes partes da cidade sendo monitorados e se deslocando, para melhor avaliar o comportamento do modelo em um ambiente real.

ACKNOWLEDGMENTS

Os autores gostariam de agradecer os seguintes órgãos de fomento: CNPq, CAPES e FAPERGS.

REFERÊNCIAS

[1] Nouf Saeed Alotaibi, Hassan I. Sayed Ahmed, Samah Osama M. Kamel, and Ghada Farouk ElKabbany. 2024. Secure Enhancement for MQTT Protocol Using Distributed Machine Learning Framework. *Sensors* 24, 5 (2024). https://doi.org/10.3390/s24051638

[2] Ali Asghari and Mohammad Karim Sohrabi. 2024. Server placement in mobile cloud computing: A comprehensive survey for edge computing, fog computing and cloudlet. *Computer Science Review* 51 (2024), 100616. https://doi.org/10.1016/j.cosrev.2023.100616

[3] Alla N. Barakat, Tariq M. Ambark, and Kenz A. Bozed. 2021. Remote Healthcare Monitoring System using IoT platform. In *The 7th Int. Conf. on Engineering & MIS 2021* (Almaty, Kazakhstan) *(ICEMIS'21)*. ACM, USA, Article 49, 5 pages. https://doi.org/10.1145/3492547.3492628

[4] Munish Bhatia, Simranpreet Kaur, and Sandeep K. Sood. 2020. IoT-Inspired Smart Toilet System for Home-Based Urine Infection Prediction. *ACM Trans. Comput. Healthcare* 1, 3, Article 14 (may 2020), 25 pages. https://doi.org/10.1145/3379506

[5] Ahmad Bilal, Zareen Tabassum, Hira Mustafa, Haseeb Khan, and Saad Umer Baig. 2024. MQTT Based Intelligent IoT Monitoring and Notification System for Enhanced Comfort of Learning Spaces. In *2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*. 1–6. https://doi.org/10.1109/KHI-HTC60760.2024.10481834

[6] Hong Chen. 2019. Ubi-Care: a Decentralized Ubiquitous Sensing Healthcare System for the Elderly Living Support. In *2019 IEEE DASC/PiCom/CBDCom/CyberSciTech*. 543–547. https://doi.org/10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00108

[7] Raffaele Chianese, Francesco Gargiulo, and Michele Arciprete. 2020. An IoT-based HIS for Healthcare Risk Management and Cost Control: A Proposal. In *Proc. of the 2019 3rd Int. Symp. on Computer Science and Intelligent Control* (Amsterdam, Netherlands) *(ISCSIC 2019)*. ACM, USA, Article 23, 6 pages. https://doi.org/10.1145/3386164.3386173

[8] Mathew Falloon, Hui Ma, and Aaron Chen. 2024. Energy-Aware Dynamic Resource Allocation and Container Migration in Cloud Servers: A Co-evolution GPHH Approach. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Melbourne, VIC, Australia) *(GECCO '24)*. Association for Computing Machinery, New York, NY, USA, 1219–1227. https://doi.org/10.1145/3638529.3654070

[9] Gabriel Souto Fischer, Gabriel de Oliveira Ramos, Cristiano André da Costa, Antonio Marcos Alberti, Dalvan Griebler, Dhananjay Singh, and Rodrigo da Rosa Righi. 2024. Multi-Hospital Management: Combining Vital Signs IoT Data and the Elasticity Technique to Support Healthcare 4.0. *IoT* 5, 2 (2024), 381–408. https://doi.org/10.3390/iot5020019

[10] Gabriel Souto Fischer, Vinicius Facco Rodrigues, Rodrigo da Rosa Righi, Cristiano André da Costa, Lucas Micol Policarpo, and Rafael Gustavo Gaspar da Silva. 2024. Looking at smart cities through the lens of a pandemic era: a systematic literature review. *International Journal of Technology Management* 94, 3-4 (2024), 342–384. https://doi.org/10.1504/IJTM.2024.136418

[11] G Geetha and K.Mohana Prasad. 2021. The Strategies to Reduce Healthcare Costs and Increase the Quality of Healthcare Service Delivery Using Hybrid Technologies. In *2021 Asian Conference on Innovation in Technology (ASIANCON)*. IEEE, Pune, India, 1–7. https://doi.org/10.1109/ASIANCON51346.2021.9544947

[12] T.H Hafsiya and Binet Rose. 2021. An IoT-Cloud Based Health Monitoring Wearable Device For Covid Patients. In *2021 7th Int. Conf. on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1. IEEE, Coimbatore, India, 266–269. https://doi.org/10.1109/ICACCS51430.2021.9441717

[13] Najwa Kouka, Vincenzo Piuri, and Pierangela Samarati. 2024. Tasks Scheduling with Load Balancing in Fog Computing: a Bi-level Multi-Objective Optimization Approach. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Melbourne, VIC, Australia) *(GECCO '24)*. Association for Computing Machinery, New York, NY, USA, 538–546. https://doi.org/10.1145/3638529.3654069

[14] Levina et al. 2023. A Novel Architecture for Developing IoT Solutions Applied to Healthcare. In *2023 2nd Int. Conf. on Applied Artificial Intelligence and Computing (ICAAIC)*. IEEE, Salem, India, 1358–1365. https://doi.org/10.1109/ICAAIC56838.2023.10140729

[15] Anwesha Mukherjee, Shreya Ghosh, Aabhas Behere, Soumya K. Ghosh, and Rajkumar Buyya. 2021. Internet of Health Things (IoHT) for personalized health care using integrated edge-fog-cloud network. *Journal of Ambient Intelligence and Humanized Computing* 12, 1 (01 Jan 2021), 943–959. https://doi.org/10.1007/s12652-020-02113-9

[16] Gautham Nagendran, R Raj, C Priya, and Harshmeet Singh. 2023. IoT Cloud Systems: A Survey. In *2023 5th Int. Conf. on Smart Systems and Inventive Technology*. IEEE, Tirunelveli, India, 415–418. https://doi.org/10.1109/ICSSIT55814.2023.10060983

[17] Vinicius Facco Rodrigues, Rodrigo Rosa Righi, Cristiano André Costa, Felipe André Zeiser, Bjoern Eskofier, Andreas Maier, and Daeyoung Kim. 2023. Digital health in smart cities: Rethinking the remote health monitoring architecture on combining edge, fog, and cloud. *Health Technol.* 13, 3 (01 Jun 2023), 449–472. https://doi.org/10.1007/s12553-023-00753-3

[18] Jorge Luiz Machado da Silva, Breno B. Nicolau de França, and Cecília Mary Fischer Rubira. 2020. Generating Trustworthiness Adaptation Plans Based on Quality Models for Cloud Platforms. In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse* (Natal, Brazil) *(SBCARS '20)*. Association for Computing Machinery, New York, NY, USA, 141–150. https://doi.org/10.1145/3425269.3425272

[19] Eleonora Tudora, Eugenia Tîrziu, Nicolae Nicolau, and Maria Gheorghe-Moisii. 2021. Fog and Cloud Computing-based IoT in Healthcare Monitoring System for Healthy Ageing. In *2021 23rd Int. Conf. on Control Systems and Computer Science (CSCS)*. IEEE, Bucharest, Romania, 489–494. https://doi.org/10.1109/CSCS52396.2021.00086

[20] Liqing Yang, Lin Chen, Lu Liu, and Haibo Chen. 2023. A Real-time, Visualization and Cloud-Linked Health Monitoring System for Disabled Elderly: Elderly Health Monitoring System. In *2022 5th Int. Conf. on Digital Medicine and Image Processing* (<conf-loc>, <city>Kyoto</city>, <country>Japan</country>, </conf-loc>). ACM, USA, 72–76. https://doi.org/10.1145/3576938.3576950

[21] Lejun Zhang, Yanfei Zou, Muhammad Hassam Yousuf, Weizheng Wang, Zilong Jin, Yansen Su, and Kim Seokhoon and. 2022. BDSS: Blockchain-based Data Sharing Scheme With Fine-grained Access Control And Permission Revocation In Medical Environment. *KSII Transactions on Internet and Information Systems* 16, 5 (May 2022), 1634–1652. https://doi.org/10.3837/tiis.2022.05.012