Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# UMA SOLUÇÃO EM NÉVOA VIA OBJETOS INTELIGENTES PARA LIDAR COM A HETEROGENEIDADE DOS DADOS EM UM AMBIENTE RESIDENCIAL

Artur H. Brandão de Souza

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Geraldo P. Rocha Filho

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# UMA SOLUÇÃO EM NÉVOA VIA OBJETOS INTELIGENTES PARA LIDAR COM A HETEROGENEIDADE DOS DADOS EM UM AMBIENTE RESIDENCIAL

Artur H. Brandão de Souza

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Geraldo P. Rocha Filho (Orientador)
CIC/UnB

Prof. Dr. Vinicius P. Gonçalves
FT/UnB

Prof. Dr. Marcos F. Caetano
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 12 de Maio de 2021

# Dedicatória

*Aos meus pais, Cláudio Henrique Gomes de Souza e Márcia Ferreira Brandão de Souza, por terem me dado todo suporte financeiro e educacional para que eu pudesse sempre estar focado em meus objetivos.*
*Às minhas avós Carmélia Gomes de Souza e Maria Zélia Ferreira Brandão, por todo carinho e apoio.*
*Aos meus irmãos, Juliana Brandão de Souza Vidal e Bruno Henrique Brandão de Souza, por terem me aturado em momentos de tensão.*
*À minha namorada, Júlia Guimarães Bernardes, por sempre estar me escutando nos momentos difíceis e me dar conselhos quando mais precisei.*

- Artur Henrique Brandão de Souza

# Agradecimentos

O autor agradece a todos que fizeram parte do desenvolvimento desse trabalho, em especial ao Prof. Dr. Geraldo Pereira por me recomendar a ideia desse projeto, além de estar sempre disposto a me auxiliar em momentos de dúvidas e me direcionar para conseguir os resultados obtidos nesse trabalho.

Ao aluno Renato Avelar e ao professor Vinicius Gonçalves por todo suporte quanto à escrita do trabalho final com importantes observações e sugestões oferecidas durante o texto.

A todos os aplicativos de música que certamente me forneceram um ambiente mais tranquilo para desenvolver a parte prática do projeto.

Aos amigos que se fizeram presente e deram apoio quando necessário: Antônio Moura, Alex Souza, Emmanuel Perotto, Igor Figueira, Khalil Carsten, Lucas Moutinho e Tiago Cabral.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

# Resumo

Com o avanço tecnológico desenfreado, o crescimento da quantidade de dispositivos e, consequentemente, da superabundância de dados é surpreendente. Tendo em vista o contexto de uma casa inteligente, para manter um controle sobre ambiente é necessário ter um grupo de dispositivos inteligentes que gerarão dados constantemente para que sejam formulados relatórios sobre o meio inserido. Devido a essa demanda, foram desenvolvidas uma série de soluções com o intuito de suprir a carência de poder de armazenamento e processamento de alguns desses equipamentos. Com isso, este trabalho propõe o HOST – uma solução que trata dos problemas da heterogeneidade de dados e da interoperabilidade de objetos inteligentes no contexto de uma casa inteligente. O HOST foi modelado com o intuito de compor um conjunto de objetos inteligentes para formar uma infraestrutura computacional em névoa. Para disseminar as informações heterogêneas, implementou-se um módulo de comunicação *Publish/Subscribe* que permite abstrair os detalhes de comunicação entre os objetos. Uma avaliação de desempenho foi realizada para validar o HOST. Os resultados mostram evidências de eficiência (i) nos recursos computacionais dos dispositivos; e (ii) na infraestrutura de comunicação. Ainda, o HOST proporciona escalabilidade no que diz respeito a quantidade de dispositivos atuando simultaneamente, além de demonstrar estar hábil a funcionar com diferentes tipos de dispositivos.

**Palavras-chave:** avanço tecnológico, casa inteligente, computação em névoa, heterogeneidade de dados, interoperabilidade, objetos inteligentes, *Publish/Subscribe*, comunicação, recursos computacionais

# Abstract

*With the fast and unstoppable technological development, the amount of available technological devices and their produced data is overwhelming. Analyzing the context of a smart home, having a diverse group of intelligent devices generating constant reports of its environment information is needed to control the house properly. Due to this demand, many possible solutions were developed in the literature to assess the need for processing power and storage capacity. This work proposes HOST – a solution that addresses the problems of data heterogeneity and the interoperability of smart objects in the context of a smart home. HOST was modeled to compose a set of intelligent objects to form a computational infrastructure in fog. To disseminate heterogeneous information, a Publish/Subscribe communication module was implemented to abstract the details of communication between objects. A performance evaluation was carried out to validate HOST. The results show evidence of efficiency (i) in the computational resources of the devices; and (ii) in the communication infrastructure. Also, HOST provides scalability about the number of devices acting simultaneously and demonstrates its ability to work with different types of devices.*

**Keywords:** *technological development, smart home, fog computing, data heterogeneity, interoperability, smart objects, Publish/Subscribe, communication, computational resources*

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Nos últimos anos, a informação se tornou um dos elementos mais valiosos do mundo [1]. Tais informações são provenientes de dados crus, observações documentadas, ou resultados de alguma medição [2]. Há diversas maneiras de se obter dados, e uma delas é por meio dos objetos inteligentes. Esses objetos podem: (i) determinar o valor de algum evento local; (ii) detectar eventos de interesse; (iii) processar tais eventos dentro ou fora da rede; (iv) detectar um objeto de interesse; e (v) prover comunicação de dados e escalabilidade.

Com os avanços tecnológicos na área de micro sistemas eletro-mecânicos, houve um crescimento significativo no uso dos objetos inteligentes dentro do contexto residencial [3]. Nesse sentido, as casas inteligentes, também conhecida como *smart home*, vêm surgindo como fonte de informação promissora e despertam o interesse de pesquisadores em propor modelos e aplicações para realizar tomada de decisões inteligentes [4]. Uma casa inteligente pode ser definida como um conjunto de objetos inteligentes presentes em uma residência para automatizar tarefas diárias e manter um controle sobre dados que o ambiente consegue prover com o intuito de gerenciar os recursos habitacionais do usuário [5].

Vale salientar que, devido a escassez de recursos computacionais presentes nos objetos inteligentes, desenvolver serviços no contexto residencial que utilizam a intercomunicação entre tais objetos não é uma tarefa trivial. Essa limitação se torna ainda mais desafiadora de ser resolvida devido a heterogeneidade dos dados e interoperabilidade entre os objetos presentes na residência. Para lidar com tais desafios, um dos caminhos promissores é o uso da computação em névoa [6] aliada com o paradigma de comunicação *Publish/Subscribe* (Pub/Sub) [7]. A computação em névoa traz o processamento e armazenamento dos dados para a borda da rede, ou seja, essas tarefas são executadas de maneira cooperativa entre os objetos inteligentes contidos no ambiente por meio da comunicação [6]. O paradigma Pub/Sub permite a comunicação confiável entres os objetos de maneira assíncrona a partir de múltiplas fontes de eventos para seus respectivos grupos de interesse [8].

Nesse contexto, a computação em névoa é caracterizada como um ambiente com um

grande número de objetos descentralizados e heterogêneos que conseguem cooperar entre si. Essa cooperação se dá através do uso da internet local que traz melhorias na performance de armazenamento e processos de tarefas sem a interferência de terceiros, como o uso da nuvem [9]. Com isso, todos os objetos que estão contidos nesse ambiente de névoa são chamados de nós de névoa, ou seja, são dispositivos que trazem processamento, armazenamento e serviços de internet para a borda da internet, também caracterizado como dispositivos fins [10]. Para que haja uma comunicação entre esses objetos pertencentes ao mesmo ambiente de névoa é utilizado o paradigma Pub/Sub que é composto por objetos que contêm características de publicadores ou subscritores que enviam e recebem mensagens, respectivamente, de maneira desacoplada e assíncrona [8].

Diferentes abordagens foram propostas para lidar com o problema da heterogeneidade dos dados e a interoperabilidade entres os dispositivos em um mesmo ambiente [11, 12, 13, 14, 15]. Apesar dos esforços significativos nessa vertente, investigar a interoperabilidade de comunicação entres os dispositivos, aproveitando os conteúdos que são disseminados de maneira implícita em um contexto residencial com base na computação em névoa é um campo de pesquisa em aberto que este trabalho investiga.

Com isso em mente, esta pesquisa apresenta HOST, uma solução para lidar como problema da HeterOgeneidade dos dados e com a interoperabilidade dos objetoS inTeligentes no contexto de uma casa inteligente. Para isso, modelou-se um ambiente computacional em névoa por meio dos objetos inteligentes providos da residencia para formar a infraestrutura do HOST. Para disseminar as informações, implementou-se nessa infraestrutura um módulo de comunicação Pub/Sub com o intuito de melhorar o custo computacional e operacional procedente da variação da quantidade de nós de névoa e da quantidade de dados trocados entre os objetos. Os resultados evidenciam que o HOST possui um desempenho satisfatório dos recursos computacionais dos dispositivos e da infraestrutura da solução proposta.

O restante desta pesquisa está organizado da seguinte maneira. O Capítulo 2 apresenta os trabalhos relacionados em que é evidenciado algumas limitações referentes a outros trabalhos que, em sua maioria, atuaram com o mecanismo de comunicação MQTT ao mesmo tempo em que o HOST consegue abranger essas restrições. Capítulo 3 descreve, em detalhes, o funcionamento do HOST e como seus componentes atuam durante o processo de execução. Já a sua validação é apresentada no Capítulo 4 enquanto Capítulo 5 apresenta as conclusões e trabalhos futuros.

# Capítulo 2

# Trabalhos Relacionados

Este capítulo abordará trabalhos relacionados a ambientes inteligentes que possuem ou contêm uma ideia similar a uma conexão MQTT para comunicação entre dispositivos. Conforme descrito no trabalho [13], os autores buscam mostrar o quão eficiente é utilizar, em um ambiente de uma casa inteligente, um MQTT broker junto com AWS (*Amazon Web Services*) para conseguir gerir uma pequena aplicação IoT (*Internet of Things*). Essa metodologia foi proposta através da implementação de ponta a ponta do cenário proposto. Assim, há um desenvolvimento de um *hardware* que captura dados de temperatura e detecção de fumaça. Esse *hardware* se conecta com o MQTT criado no AWS e há o desenvolvimento de um aplicativo para a captação dos eventos transmitidos pelo MQTT. Portanto, esse trabalho conseguiu demonstrar uma grande facilidade que se há em desenvolver uma aplicação MQTT no AWS para pequenas aplicações *IoTs*. Contudo, há uma limitação quanto a ideia de desempenho com o uso dessa aplicação, em que, em um provável aumento da quantidade de dispositivos poderia acarretar em um problema de desempenho do *broker*.

Em um outro trabalho [14] é mostrado uma forma de lidar com uma grande quantidade de dados heterogêneos captados por dispositivos *IoT* utilizando uma ferramenta para armazenamento em nuvem chamada *Apache Kafka*. Para isso, desenvolveram uma interface *webservice* REST que busca facilitar o uso da plataforma utilizada e o *cloud* da Google como forma de armazenamento. Como resultado, as métricas calculadas para medir o desempenho de toda a transmissão das mensagens até a nuvem foram consideradas como eficientes, levando como conclusão a possibilidade do uso dessa arquitetura para a transmissão de dados em tempo real. Por fim, foram apenas analisados formas de enviar os dados diretamente para a nuvem, podendo ter expandido o estudo para a comunicação em névoa entre os dispositivos locais.

Tomando como base o artigo [12], foi proposto a criação de um *MQTT broker clustring* visando a escalabilidade e o baixo custo através de uma grande quantidade de dispositivos

IoTs conectados em um mesmo ambiente. Para este propósito, foi criado um sistema de visualização com a ideia de analisar os dados do *cluster* quanto ao uso da CPU, memória e o tráfico de rede. Como forma de simular a proposta, foram utilizados quinhentos dispositivos Raspberry Pi como nós e para a comunicação de mensagens entre eles é utilizado o *broker* MQTT através do Mosquitto, além do uso do *docker* e *NGINX*. Ademais, a escalabilidade é adquirida através do uso de contêineres do *docker* e, com o *NGINX* sendo aplicado no *backend* dos MQTT *brokers*, trouxe um resultado positivo quanto as métricas avaliadas encontradas nas analises das visualizações. Uma forma de melhorar a avaliação seria abranger a quantidade de protocolos MQTT como troca de mensagens, visando comparar as métricas de de forma mais detalhada.

Com a proposta de gerenciar aplicações de uma residência através de um ambiente computacional de *fog*, o artigo [16] traz como solução o uso de inteligência artificial. Assim, essa solução tem como objetivo garantir uma rápida tomada de decisão ao executar o sistema localmente na infraestrutura desenvolvida pelos autores. Com isso, a disseminação de dados internamente foi feita através dos métodos Pub/Sub o que permitiu a viabilidade de escalabilidade de dispositivos sem causar problemas de comunicação entre novos dispositivos. Como resultado, o ImPeRIum (nome dado a arquitetura proposta) conseguiu viabilidade e uma eficiência melhor que outras literaturas quando se trata de ambientes que contenham dispositivos com recursos escassos. Levando em conta a configurações internas dentro de protocolos MQTT, utilizado no trabalho como meio de comunicação entre os dispositivos, haveria uma possibilidade de acrescentar estudos quanto ao aumento do tamanho das mensagens e analisar se há uma interferência quanto as métricas avaliadas.

Na busca busca de um sistema de *smart home* com um custo baixo e flexível, o [11] demonstra uma aplicação para Android que comunica através de um *micro-web server* para prover diversas funcionalidades de interação entre os dispositivos. Para isso, foram utilizados *Arduino Ethernet* como forma de conectar sensores com a ideia de capturar a temperatura, umidade e sensibilidade de gás no ambiente. Assim, esses dispositivos fazem uma comunicação com a aplicação desenvolvida, em que essa aplicação tem o objetivo de ter um controle sobre esses sensores e capturar os dados informados. Através de testes, a aplicação conseguiu chegar no resultado esperado, demonstrando uma viabilidade e eficácia na comunicação de dispositivos *smart home* com aplicações *Android*. No entanto, como houve apenas verificação quanto ao sucesso na conectividade da aplicação, poderia ser verificado outros meios de comunicações entre dispositivos, como MQTT. Com essa comunicação, poderia trazer uma velocidade maior quanto a transferência de dados e facilidade na manutenção do sistema proposto devido a grande capacidade de escalabilidade e operatividade do seu uso perante a outros meios de comunicação demonstrado em *surveys*

como [17].

Já no artigo [18] é feito uma comparação do uso da computação em nuvem e em névoa quanto a métricas que dizem respeito ao tráfego dos dados nesses dois tipos de ambientes. Com isso, esse artigo traz dados de latência e atrasos causados devido a transferência de dados, o que, dependendo do ambiente, é visto que interfere bastante quanto a performance dessa comunicação. Com isso, através dos dados coletados de ambos ambientes, é notório um grande aumento no desempenho no ambiente em névoa levando em conta a busca de baixa latência e evitar congestionamentos devido ao tráfego de internet. Assim, há ainda uma necessidade de avaliar métodos de comunicação dentro da própria configuração em névoa, com a ideia de comparar as mesmas métricas.

A Tabela 2.1 apresenta as limitações supracitadas em cada um dos trabalhos citados, comparando-os com esta pesquisa. A grande maioria dos trabalhos que fazem o uso do paradigma Pub/Sub não fazem menção ao conceito de computação em névoa. Por outro lado, os trabalhos que não aderiram ao uso desse paradigma, possuem fatores relacionados com o conceito de computação em névoa. Além disso, ainda há estudos que oferecem um foco maior na aplicação desenvolvida sem ponderar um possível aumento de objetos em seus projetos e o quanto isso poderia afetar as abordagens propostas.. Assim, as abordagens apresentadas na Tabela 2.1 são: a utilização do paradigma Pub/Sub, a utilização da computação em névoa e se contém o estudo de escalabilidade sobre o que é proposto. Para os pontos que contêm ausência de um desses tópicos, os campos estarão em branco enquanto os que tiverem um símbolo no campo, esses obtiveram alguma citação ou abordam de forma completa os temas.

| **Trabalho Relacionado** | **Pub/Sub** | **Computação Em Névoa** | **Escalabilidade** |
|---|---|---|---|
| Kang *et al.* 2017 [13] | ✓ | | |
| de Sousa *et al.* 2018 [14] | ✓ | | ✓ |
| Jutadhamakorn *et al.* 2017 [12] | ✓ | | ✓ |
| Rocha Filho *et al.* 2018a [16] | ✓ | ✓ | ✓ |
| Kumar 2014 [11] | | ✓ | |
| Dastjerdi *et al.* 2016 [18] | | ✓ | ✓ |
| HOST | ✓ | ✓ | ✓ |

Tabela 2.1: Comparação das limitações dos trabalhos relacionados em relação ao HOST.

# Capítulo 3

# Uma solução em névoa para uma casa inteligente

Este capítulo apresenta o HOST [19], uma solução em névoa para lidar com o problema da heterogeneidade dos dados e com a interoperabilidade dos objetos inteligentes no contexto residencial. O HOST foi desenvolvido com o intuito de ter um conjunto de objetos inteligentes providos pela própria residencia, os quais estão conectados à internet para formar um ambiente computacional em névoa. Esse ambiente, também chamado de computação em borda, gera uma cooperação entre o uso da internet e os dispositivos heterogêneos, que trabalham juntos para prover uma melhora na performance de armazenamento e no processamento de tarefas por estarem na camada de borda da rede [6]. Uma vez que o próprio conceito da computação em nuvem ratifica seus limites [20], a computação em névoa passa a ser uma opção praticável ao entregar uma baixa latência e uma melhora na taxa de resposta entre os objetos que compõem esse ambiente.

A partir desse espaço formado no HOST, é implementado um módulo que abstrai os detalhes de comunicação dos objetos inteligentes com base no módulo de comunicação Pub/Sub. Devido ao desacoplamento oferecido entre as partes finais por meio do Pub/Sub, é possível criar grupos de interesses para cada dado coletado. Nesse sentido, é factível garantir que os módulos de ambiente, objetos e aplicações possam comunicar entre si, auxiliando no processo de disseminação de dados, como apresentado na Figura 3.1.

O HOST contém objetos inteligentes que produzem dados e os enviam para o módulo de ambiente através da conexão MQTT Pub/Sub, como apresentado na Figura 3.1. Além disso, as aplicações atuam em cima desses dados coletados no ambiente. Para tanto, a infraestrutura de funcionamento do ambiente do HOST foi dividida em duas, sendo elas: (i) névoa; e (ii) nuvem. No ambiente em névoa, o armazenamento e o processamento das tarefas são realizadas localmente por meio dos nós de névoa. Por outro lado, o ambiente em nuvem é um servidor remoto com maior recurso computacional que, por exemplo,

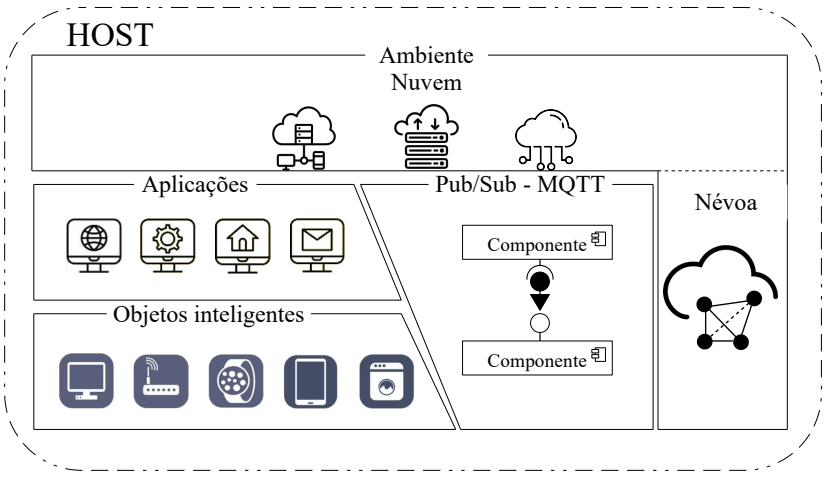



Figura 3.1: Visão geral do HOST

armazena dados históricos e processa tarefas que requerem um alto custo tecnológico. Em síntese, o HOST tem como objetivo principal garantir eficiência na comunicação para a transferência de dados entre dispositivos em um mesmo ambiente, ao mesmo tempo em que consegue prover eficiência nos recursos computacionais limitados dos objetos sem que haja problemas de escalabilidade nos processos executados. Para um melhor entendimento do HOST, a Seção 3.1 apresenta o ambiente em névoa adotado. Em seguida, a Seção 3.2 apresenta o mecanismo de comunicação utilizado. Por fim, são apresentadas aplicações práticas do HOST, Seção 3.3.

## 3.1 Ambiente computacional em névoa

No ambiente computacional em névoa, assume-se que cada objeto inteligente é equipado com capacidade de armazenamento, poder de processamento e interface de comunicação sem fio. No HOST, os objetos inteligentes são chamados de nós de névoa que são responsáveis por coletar, processar e disseminar os dados no ambiente residencial, como visto na Figura 3.2. Ainda, esses objetos estão na borda da rede e são responsáveis por realizar uma comunicação entre os usuários e o servidor em nuvem. O objetivo do conjunto dos nós de névoa é formar um ambiente computacional capaz de realizar tomada de decisões em uma área que necessita enviar dados com uma performance superior a que um ambiente em nuvem consegue alcançar. Salienta-se que não é o escopo deste trabalho propor modelos e aplicações para realizar tomada de decisões, mas propor uma solução que possa ser utilizada para isso.

Os nós de névoa estão distribuídos no ambiente e atuam como produtor e consumidor de informações. Cada equipamento inteligente possui o conhecimento local e global

do ambiente. Para isso, o HOST adota um mapa virtual de névoa com base no trabalho [16]. Para gerar o mapa, o nó de névoa compartilha a informação mediante um modelo *request-reply* do Pub/Sub. O compartilhamento é feito via um mecanismo de comunicação implementado no HOST que será apresentado a seguir.

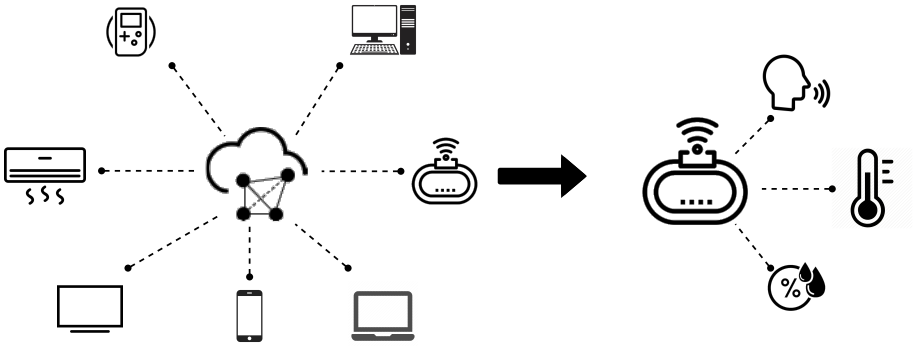


Figura 3.2: Visão geral do ambiente em névoa e seus respectivos nós

## 3.2 Mecanismo de comunicação

A solução proposta foi desenvolvida com base no protocolo MQTT (*Message Queue Telemetry Transport*) [8] no ambiente em névoa. Esse protocolo contém conceitos básicos que são compostos por: (i) Publicador/Subscritor, são dispositivos que enviam e recebem mensagens, respectivamente, de maneira desacoplada e assíncrona entre os dispositivos [8]; (ii) tópicos e subscrições, os tópicos podem ser relacionados como o assunto da mensagem enviada pelo publicador enquanto os subscritores se subscrevem nos tópicos para que recebam as mensagens apenas referentes a esses tópicos; (iii) qualidade do serviço, consiste na garantia de entrega das mensagens e são divididos em QoS0 (no máximo uma vez), QoS1 (pelo menos uma vez), QoS2 (exatamente uma vez); (iv) retenção de mensagens, as mensagens permanecem no *broker* para que futuros subscritores do tópico em questão recebam as mensagens [21].

Além disso, Para o funcionamento desse protocolo são exigidos dois componentes em sua arquitetura, que são compostos por: (i) clientes, que funcionam como qualquer objeto que atue como publicador ou subscritor; (ii) *broker*, atua como um controle de distribuição das mensagens enviadas pelos clientes em que coleta mensagens enviadas pelo publicador e redireciona-as para os subscritores interessados nos tópicos das mensagens [21].

Com isso, a escolha pelo MQTT é condicionada devido a miníma utilização da largura de banda, da baixa imposição de recursos sobre equipamentos que ainda assim conseguem transmitir com uma confiabilidade e garantia de entrega das mensagens enviadas sobre ele, além do desacoplamento entre os componentes que fazem parte do ambiente,

características presentes nesta pesquisa. Assim, há três componentes principais no HOST atuando no processo do MQTT, Figura 3.3, sendo eles: (i) publicador, componente que envia dados por meio de um evento; (ii) evento, componente que gera uma notificação; e (iii) assinante, componente que recebe dados de interesse sobre um evento.

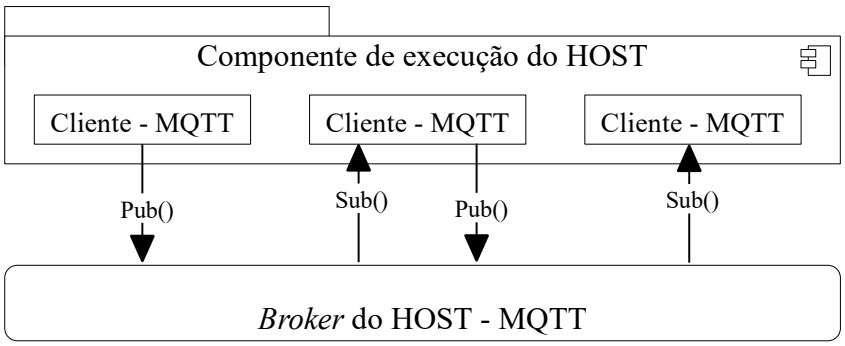



Figura 3.3: Protocolo de comunicação Pub/Sub

No HOST, o desacoplamento que há entre os objetos é uma dissociação de espaço, tempo e de sincronização. Quanto ao espaço, é relacionado com a não necessidade dos nós saberem sobre a quantidade de publicadores e subscritores pertencentes à névoa, uma vez que há uma independência na atividade de cada equipamento que faz o uso do MQTT. Além disso, não há necessidade dos publicadores e subscritores estarem ativos ao mesmo tempo para que haja uma troca de mensagens entre si. Com isso, ao publicar um evento, mesmo que o assinante esteja inativo no momento que é gerado a notificação, este poderá ser acionado e não há necessidade do publicador estar ativo para que o evento seja direcionado a seu assinante, uma vez que quem faz essa função é o *broker* do MQTT no HOST. Além disso, o HOST por meio do MQTT traz aos publicadores a possibilidade de produzirem eventos sem a necessidade de que alguns deles sejam bloqueados ou impedidos de enviar mensagens. De maneira semelhante ocorre com os assinantes, quanto ao recebimento desses eventos, em que, o *broker* está encarregado de garantir a remessa das mensagens para os respectivos subscritores. Em razão disso, a solução descrita consegue trazer facilidades através da execução assíncrona de suas atividades, o que possibilita realizar tarefas concorrentes entre todos os nós de névoa presentes no ambiente para lidar com o problema da heterogeneidade dos dados e interoperabilidade entres os nós.

Para o manuseio dos eventos, o HOST utiliza o *broker* da biblioteca *Eclipse Paho MQTT*, que constitui em um componente do protocolo *MQTT*, revisite a Figura 3.3. A biblioteca trabalha na captura dos dados enviados pelos publicadores, gera uma notificação e a direciona para seus respectivos assinantes. Para que isso ocorra, tanto os publicadores quanto os subscritores utilizam os comandos *publish()* e *subscribe()*, respectivamente, ao importar a biblioteca para que consigam se comunicar com o *broker*.

## 3.3 Aplicabilidade

A partir do desenvolvimento do HOST, é possível criar diversas aplicações e modelos para tomada de decisões que podem ser empregadas em ambientes inteligentes. Essa solução pode ser utilizada para aplicações de monitoramento e controle do ambiente com a ideia de realizar processos de tomada de decisões para cuidados de idosos em uma *Health Smart Homes*. O cenário supramencionado utiliza o HOST para coletar dados heterogêneos como dados de voz, temperatura, umidade, vídeo, e termômetro corporal. A partir dessa coleta, cada dispositivo *Pub* enviaria, para o *Sub* de interesse, o tópico ao qual se refere o dado e a mensagem registrada. Com isso, avaliações em tempo real seriam realizadas, mantendo, por exemplo, um controle sobre o bem estar do usuário.

Outra aplicação que o HOST pode oferecer seria a verificação constante no sistema de resfriamento em shoppings. Isso ocorreria devido a otimização da transmissão de dados em razão do ambiente em névoa entre os sensores de temperaturas do ambiente, como mostrado no HOST. Com o objetivo de manter uma temperatura agradável a todos os clientes, seria possível ter um controle onde há um conglomerado de pessoas em regiões do shopping através do aumento da temperatura ambiente que seria ocasionada pela temperatura corporal das pessoas presentes nessas regiões. Com isso, poderia aumentar a potência do resfriamento nesses locais. Por outro lado, a partir desses dados coletados, seria possível também buscar uma economia de energia pelo próprio shopping. Essa economia se daria devido a redução da potência do resfriamento nos locais em que não haja detecção de grande quantidade de pessoas, o que levaria a manter a temperatura local.

Além disso, levando em conta o contexto socio-politico atual e a crise ocasionada pelo Coronavirus (COVID-19), haveria uma possibilidade de identificar aglomerações através de sensores de presença ou até mesmo de temperatura. No momento em que identificasse um acumulado grupo de pessoas, os responsáveis pelo local poderão ser notificados sobre a ocorrência e, consequentemente, tomar as devidas precauções.

# Capítulo 4

# Resultados Experimentais

Este capítulo apresenta o cenário modelado bem como os resultados gerados na avaliação de desempenho do HOST. Para tanto, o HOST foi validado em duas etapas, sendo elas: (i) avaliação dos recursos computacionais da infraestrutura; e (ii) avaliação da disseminação dos dados na rede. Por meio dessas avaliações, foi possível entender o funcionamento do HOST e a sua vantagem em um ambiente de fog. A seguir, será apresentado o cenário modelado, os parâmetros selecionados e as métricas utilizadas para gerar os resultados.

## 4.1 Cenário

Para avaliar a proposta, utilizou-se o contêiner do *docker* como um ambiente para simular os dispositivos Pub/Sub em que tais dispositivos representam os nós de névoa no HOST. A linguagem utilizada foi o Python 3.6 com a biblioteca *paho-mqtt* [22] para atuar como *broker*. Como métricas utilizadas para avaliar os recursos computacionais do HOST e computar a disseminação dos dados na rede, destacam-se: (i) **uso do disco** pelo sistema operacional; (ii) **informações da memória**, que consistem na soma da memória física não trocada com o total de memória virtual utilizada pelo processo; (iii) **tempo gasto da cpu** que representa a soma do tempo do usuário com o tempo em modo *kernel* pelo processo; (iv) **tempo de resposta** para enviar um tópico; (v) **taxa de dados** entre o dispositivo e infraestrutura; e (vi) **taxa de entrega** por segundo dos tópicos na infraestrutura. Para obter tais métricas, foi utilizada a biblioteca *psutils* [23] no *Python* que tem como objetivo coletar informações dos processos e do próprio sistema operacional.

Na geração dos resultados, foi realizada uma variação do número de nós de névoa (5, 10, 20 e 30), e uma variação do número de tópicos (8, 13, 21 e 34) para investigar o impacto no HOST, como visto na Tabela 4.1. Neste caso, os nós de névoa, publicadores, produziram simultaneamente seus próprios dados, enquanto um subscritor recebe por meio do *broker*. Ainda, variou-se o *payload* dos tópicos (100kB, 200kB, 400kB, 800kB

e 1600kB) instanciando um *broker* e os dados foram normalizados quanto ao tempo de resposta (ms) e tráfego de dados (kB/s). Cada experimento foi executado 32 vezes com um intervalo de confiança de 95% usando a distribuição *t-student*, como apresentado nas subseções a seguir.

| **Parâmetro** | **Valor padrão** |
|---|---|
| Número de Tópicos | 8, 13, 21 e 34 |
| Número de nós de névoa | 5, 10, 20 e 30 |
| Payload | 100kB, 200kB, 400kB, 800kB e 1600kB |
| Broker | 1 |
| Protocolo | Eclipse Paho MQTT |
| Número de replicações | 32 |
| Intervalo de confiança | 95% |

Tabela 4.1: Conjunto de parâmetros utilizados para gerar os resultados.

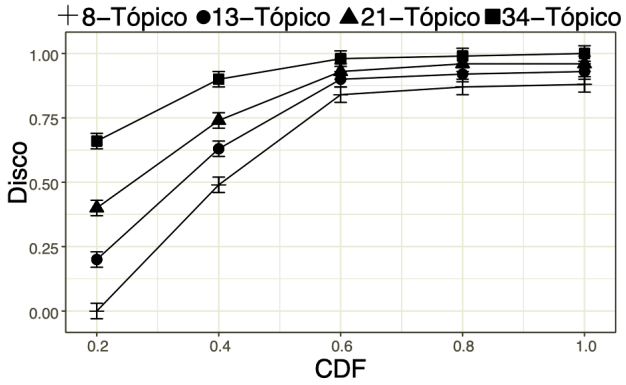
## 4.2 Impacto dos recursos computacionais dos dispositivos no HOST

Na Figura 4.1, é apresentada a avaliação de desempenho dos recursos computacionais do HOST variando a quantidade de tópicos. Inicialmente, foi avaliado o uso do disco do sistema operacional em função do CDF (*Cumulative Distribution Function*), como apresentado na Figura 4.1a. Independentemente da quantidade de tópicos enviados, observa-se que o uso do disco possui estabilidade no seu comportamento, além de uma escalabilidade no seu uso à medida que aumenta a quantidade recursos utilizados no HOST. Isso faz sentido uma vez que a solução proposta aproveita o processamento local provido do ambiente computacional em névoa. Assim, é possível aumentar a quantidade de tópicos sem se preocupar em um crescimento do uso do disco que pudesse afetar o publicador.

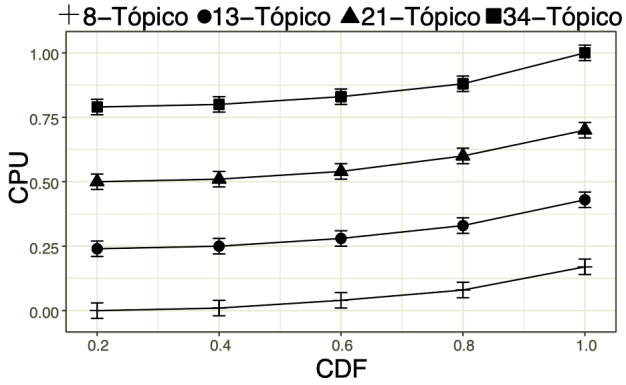
Após avaliar o uso do disco, analisou-se o tempo gasto de cpu para executar o processo, como apresentado na Figura 4.1b. Por meio dos resultados, observa-se que durante todo o processo há um crescimento suave em função do tempo de execução do usuário. Esse comportamento pode ser explicado devido ao Pub/Sub não enviar os tópicos diretamente para os dispositivos. Isto é, apenas os assinantes que manifestam interesse podem receber o tópico enviado. Consequentemente, não haverá um crescimento inesperado do uso da cpu na solução proposta.

Em relação a métrica de memória, nota-se que não há uma diferença no comportamento da métrica independentemente da quantidade de tópicos, como apresentado na Figura 4.1c. Em outras palavras, a memória física e virtual utilizada pelo processo per-

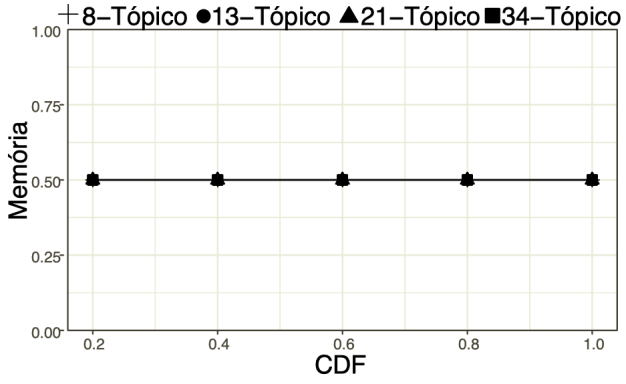
manece inalterada independente do aumento de tópicos enviados. Portanto, a quantidade de memória a ser utilizada para o publicador pode ser próxima a constante que ela atinge durante sua execução, não necessitando de buscar alternativas caso venha a necessitar alterar a quantidade de envio tópicos.




(a) Disco




(b) CPU

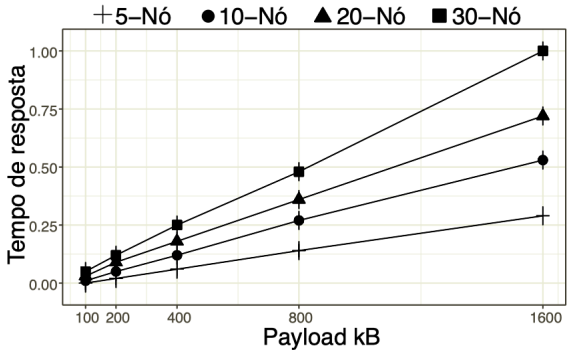



(c) Memória

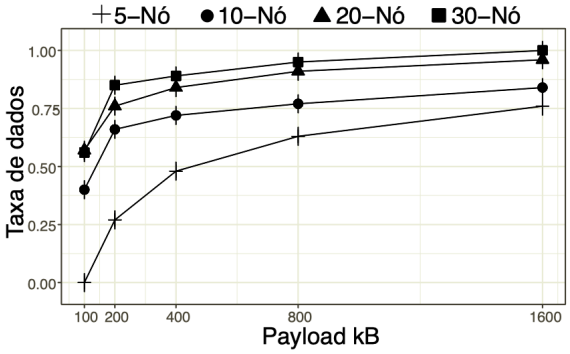
Figura 4.1: Impacto de desempenho do uso do disco, tempo gasto de CPU e memória.

## 4.3 Impacto dos recursos computacionais da infraestrutura do HOST

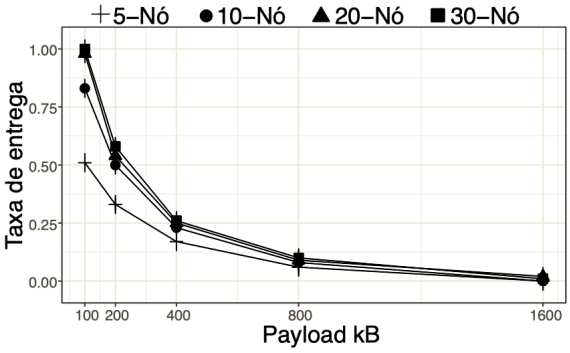
Avaliamos o desempenho do HOST quanto ao aumento da quantidade de nós de névoa em função do *payload* enviados por eles. Conforme a quantidade de *payload* aumenta, observa-se que há um aumento no tempo de reposta para enviar a mensagem, como apresentado na Figura 4.2a. Isso está relacionado com o *broker*, o qual necessita de um tempo maior de retorno devido a captação e envio das mensagens.




(a) Tempo de resposta




(b) Taxa de Dados




(c) Taxa de entrega de mensagens

Figura 4.2: Impacto do tempo de resposta, taxa de dados e entrega de mensagens.

Quanto a taxa de dados, analisada na Figura 4.2b, nota-se uma estabilidade no crescimento após 400 kB de *payload*. Esse fato ratifica que é possível aumentar a quantidade de dispositivos sem sobrecarregar a infraestrutura do HOST. Além disso, a avaliação corrobora o fato da não utilização de memória pelo *broker*, como apresentado na Figura 4.1c e, assim, não gerando um obstáculo para o tráfego de dados que ocorre durante o processo. Vale salientar que com o aumento da quantidade de dispositivos em função do *payload*, a taxa de entrega de mensagens diminui, como observado na Figura 4.2c. Esse resultado advém do aumento da taxa de dados que o *broker* consegue recebê-los e repassar para os demais subscritos nos tópicos em questão. Contudo, mesmo que a taxa de entrega diminua conforme o *payload* aumenta, é notório que a quantidade de nós não afeta de modo significativo o resultado, chegando a uma estabilidade da taxa de entrega. Portanto, o HOST além de ser eficiente quanto a taxa de dados, também demonstra bons resultados quanto ao tempo de resposta e entrega de mensagens com o aumento de *payload*.

# Capítulo 5

# Conclusão

Esta pesquisa aborda o problema da heterogeneidade de dados e a interoperabilidade dos objetos inteligentes no contexto residencial. Como resultado deste trabalho propôs o HOST, uma infraestrutura que abstrai os detalhes de comunicação de objetos inteligentes com base no módulo Pub/Sub. A avaliação de desempenho apresenta evidencias da eficiência nos recursos computacionais dos dispositivos e da infraestrutura de comunicação do HOST. Os resultados mostraram que o HOST proporciona escalabilidade no que diz respeito quantidade de dispositivos atuando simultaneamente, além de demonstrar estar hábil a funcionar com diferentes tipos de dispositivos e dados.

Os principais resultados obtidos estão relacionados com a fácil previsão do custo operacional e computacional do ambiente em névoa. Isso ocorre tendo em vista a estabilidade e padrões nas projeções dos resultados referentes ao uso do disco, tempo de CPU, uso de memória, além do desempenho do tempo de resposta, taxa de dados e entrega de mensagens.

## 5.1 Trabalhos Futuros

Como trabalhos futuros, planeja-se expandir os estudos desse artigo ao procurar executar o HOST com dados heterogêneos e trabalhar com publicadores e subscritores distintos quanto a linguagem de programação e infraestrutura. Da mesma forma, expandir o estudo de resultado para outras métricas e ampliar o HOST para diferentes contextos voltados para ambientes inteligentes. Além disso, pensando em um contexto de segurança dos dados, a ampliação desse trabalho sobre esse contexto é de suma importância devido aos dados capturados pelo ambiente em névoa residencial terem um caráter pessoal que apenas deve ser compartilhado a partir da autorização dos próprios usuários.

Por outro lado, trabalhar com o uso de dispositivos inteligentes reais, como um Raspberry Pi devido a sua limitação computacional e de armazenamento que é o foco dessa

pesquisa. Ademais, um estudo mais aprofundado quanto a melhorar o desempenho do *broker* através do uso de algumas plataformas como o Apache Kafka [24].

# Bibliografia


[1] M. S. A. Vitorino, L. C. B. Moura e H. J. S. R. Cosenza. "Sociedade da Informação, quando os dados pessoais viram moeda de troca: A polêmica da nova moeda". Em: (2018).

[2] João Luiz Becker. *Estatıstica básica: transformando dados em informação*. Bookman, 2015.

[3] Vinıcius P Gonçalves, PR Geraldo Filho, Leandro Y Mano e Rodrigo Bonacin. "FlexPersonas: flexible design of IoT-based home healthcare systems targeted at the older adults". Em: *AI & SOCIETY* (2021), pp. 1–19.

[4] Michal Luria, Guy Hoffman e Oren Zuckerman. "Comparing Social Robot, Screen and Voice Interfaces for Smart-Home Control". Em: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 580–628. ISBN: 9781450346559. DOI: 10.1145/3025453.3025786. URL: https://doi.org/10.1145/3025453.3025786.

[5] Hongbo Jiang, Chao Cai, Xiaoqiang Ma, Yang Yang e Jiangchuan Liu. "Smart home based on WiFi sensing: A survey". Em: *IEEE Access* 6 (2018), pp. 13317–13325.

[6] Shanhe Yi, Cheng Li e Qun Li. "A Survey of Fog Computing: Concepts, Applications and Issues". Em: *Proceedings of the 2015 Workshop on Mobile Big Data*. Mobidata '15. Hangzhou, China: Association for Computing Machinery, 2015, pp. 37–42. ISBN: 9781450335249. DOI: 10.1145/2757384.2757397. URL: https://doi.org/10.1145/2757384.2757397.

[7] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui e Anne-Marie Kermarrec. "The many faces of publish/subscribe". Em: *ACM computing surveys (CSUR)* 35.2 (2003), pp. 114–131.

[8] Ying Liu, Beth Plale et al. "Survey of publish subscribe event systems". Em: *Computer Science Dept, Indian University* 16 (2003).

[9] Luis M. Vaquero e Luis Rodero-Merino. “Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing”. Em: *SIGCOMM Comput. Commun. Rev.* 44.5 (out. de 2014), pp. 27–32. ISSN: 0146-4833. DOI: 10.1145/2677046.2677052. URL: https://doi.org/10.1145/2677046.2677052.

[10] Hany F. Atlam, Robert J. Walters e Gary B. Wills. “Fog Computing and the Internet of Things: A Review”. Em: *Big Data and Cognitive Computing* 2.2 (2018). ISSN: 2504-2289. DOI: 10.3390/bdcc2020010. URL: https://www.mdpi.com/2504-2289/2/2/10.

[11] Shiu Kumar. “Ubiquitous smart home system using android application”. Em: *arXiv preprint arXiv:1402.2114* (2014).

[12] Pongnapat Jutadhamakorn, Tinnapat Pillavas, Vasaka Visoottiviseth, Ryousei Takano, Jason Haga e Dylan Kobayashi. “A scalable and low-cost MQTT broker clustering system”. Em: *2017 2nd International Conference on Information Technology (INCIT)*. IEEE. 2017, pp. 1–5.

[13] Do-Hun Kang, Min-Sung Park, Hyoung-Sub Kim, Da-young Kim, Sang-Hui Kim, Hyeon-Ju Son e Sang-Gon Lee. “Room temperature control and fire alarm/suppression IoT service using MQTT on AWS”. Em: *2017 International Conference on Platform Technology and Service*. IEEE. 2017.

[14] Adriano B. de Sousa, José R. Torres Neto, Geraldo P. R. Filho e Jó Ueyama. “Uma plataforma de IoT para integração de dispositivos baseada em nuvem com Apache Kafka”. Em: *Anais Estendidos do XXXVI SBRC*. Campos do Jordão: SBC, 2018. URL: https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/2504.

[15] GP Rocha Filho, L. Yukio Mano, A. Demetrius Baria Valejo, L. Aparecido Villas e J. Ueyama. “A Low-Cost Smart Home Automation to Enhance Decision-Making based on Fog Computing and Computational Intelligence”. Em: *IEEE Latin America Transactions* 16.1 (2018), pp. 186–191. DOI: 10.1109/TLA.2018.8291472.

[16] GP Rocha Filho, José R Torres Neto, Alan Valejo, Rodolfo I Meneguette, Leandro A Villas e Jó Ueyama. “Um sistema de controle neuro-fog para infraestruturas residenciais via objetos inteligentes”. Em: *Anais do XXXVI SBRC*. SBC. 2018.

[17] Biswajeeban Mishra e Attila Kertesz. “The Use of MQTT in M2M and IoT Systems: A Survey”. Em: *IEEE Access* 8 (2020), pp. 201071–201086.

[18] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh e Rajkumar Buyya. “Fog computing: Principles, architectures, and applications”. Em: *Internet of things*. Elsevier, 2016, pp. 61–75.

[19] Artur Henrique B. de Souza. *HOST Implementação do HOST*. 2021. URL: https://github.com/arturhbs/HOST (acesso em 21/05/2021).

[20] Niroshinie Fernando, Seng W. Loke e Wenny Rahayu. "Mobile cloud computing: A survey". Em: *Future Generation Computer Systems* 29 (2013). Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures, pp. 84–106. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2012.05.023. URL: https://www.sciencedirect.com/science/article/pii/S0167739X12001318.

[21] Dipa Soni e Ashwin Makwana. "A survey on mqtt: a protocol of internet of things (iot)". Em: *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*. Vol. 20. 2017.

[22] *paho-mqtt Broker MQTT*. 2021. URL: https://pypi.org/project/paho-mqtt/ (acesso em 21/05/2021).

[23] *psutil Biblioteca para acessar dados do processo e sistema operacional*. 2021. URL: https://pypi.org/project/psutil/ (acesso em 21/05/2021).

[24] *Apache Kafka Plataforma Apache Kafka*. 2021. URL: https://kafka.apache.org/ (acesso em 21/05/2021).