UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANDERSON ALTAIR TOMKELSKI

FOG COMPUTING: ESTUDO DE CASO USANDO AMAZON WEB SERVICES

CHAPECÓ
2019

ANDERSON ALTAIR TOMKELSKI

FOG COMPUTING: ESTUDO DE CASO USANDO AMAZON WEB SERVICES

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Marco Aurélio Spohn

CHAPECÓ
2019

Tomkelski, Anderson Altair

FOG COMPUTING: ESTUDO DE CASO USANDO AMAZON WEB SERVICES / Anderson Altair Tomkelski. – 2019.
36 f.: il.

Orientador: Marco Aurélio Spohn.
Trabalho de conclusão de curso (graduação) – Universidade Federal da Fronteira Sul, curso de Ciência da Computação, Chapecó, SC, 2019.

1. Fog computing. 2. Cloud computing. 3. Internet of Things. 4. Amazon Web Services. 5. ESP32. I. Spohn, Marco Aurélio, orientador. II. Universidade Federal da Fronteira Sul. III. Título.

E-mail: ander.tomkelski@gmail.com

ANDERSON ALTAIR TOMKELSKI

FOG COMPUTING: ESTUDO DE CASO USANDO AMAZON WEB SERVICES

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Marco Aurélio Spohn

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em: 04/12/2019.

BANCA AVALIADORA

Marco Aurélio Spohn – UFFS

Braulio Adriano De Mello – UFFS

Emílio Wuerges – UFFS

AGRADECIMENTOS

Inúmeras pessoas merecem agradecimento na finalização dessa jornada. Primeiramente, agradeço todo o apoio e incentivo de minha família em relação aos meus estudo e minha vida. Agradeço também a todos os professores que tive contato durante a graduação em especial ao Prof. Dr. Marco Aurélio Spohn pela orientação nesse trabalho. Por fim agradeço a todos meus colegas que tive o privilégio de conviver durante o curso e a todos meus amigos que ajudaram a tornar essa jornada mais divertida e menos cansativa.

"Nos desculpamos pelo inconveniente"
(Douglas Adams, Até mais, e Obrigado pelos Peixes!)

RESUMO

O presente trabalho possui como objetivo avaliar os recursos disponíveis e desempenho na utilização de um ecossistema *Fog Computing* para internet das coisas. Para realização desse trabalho foi utilizado os serviços disponíveis pela *Amazon Web Services* como suporte ao estudo de caso e seus recursos foram analisados e testados como soluções. Uma discussão é feita sobre os recursos avaliados e sobre os resultados obtidos bem como a utilização do ecossistema *Fog Computing* em contrapartida a *Cloud Computing*.

Palavras-chave: Fog computing. Cloud computing. Internet of Things. Amazon Web Services. ESP32.

ABSTRACT

The present work has as objective to evaluate the available resources and performance in the use of Fog Computing ecosystem for internet of things. To perform this work, the services available through Amazon Web Services were used to support the case study and their resources were analyzed and tested as solutions. A discussion is made about the resources evaluated and the results obtained as well as the use of Fog Computing ecosystem as opposed to Cloud Computing.

Keywords: Fog computing. Cloud computing. Internet of Things. Amazon Web Services. ESP32.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A computação, controle e armazenamento de dados em nuvem se tornou uma tendência. A nuvem se tornou uma maneira fácil de acessar os dados e obter processamento. Desse modo, os dispositivos que operam as aplicações diretamente não precisam ter um poder de armazenamento e processamento tão elevado. A interação direta com data centers que possuem grande poder computacional permitiu que uma série de novas aplicações fossem criadas. Assim, o conceito de internet das coisas tem se difundido com mais facilidade, pois a interação com a nuvem permite que sensores e microcontroladores captem informações e apenas enviem-nas para o data center. Ao receber a informação das aplicações, o data center precisa armazenar ou processar tal informação (9).

Muitos desafios são encontrados nessa comunicação direta com a nuvem, tornando necessário que uma nova arquitetura fosse pensada para trabalhar mais próximo dos clientes finais. Entre os principais problemas encontrados no modelo de computação em nuvem se destacam: requisitos de latência rigorosos, onde o tempo de resposta deve ser previsto; restrições de largura de banda, em que a quantidade de dados gerados pelas aplicações pode ser muito grande; recursos limitados aos dispositivos, onde além de todos os protocolos que os dispositivos que atendem as aplicações devem suprir, não é garantido que esses dispositivos tenham capacidade de processar operações sofisticadas, as quais podem ocorrer de maneiras extraordinárias; garantia de ininterrupção de serviço, no qual deve-se garantir que a aplicação não perderá a conexão com a nuvem e existe uma dificuldade muito grande de prover ininterrupção de serviço com ela. Além dos desafios já citados, há os problemas encontrados no quesito de segurança e atualização desses dispositivos (9).

Para resolver essa série de problemas apresentados, uma nova arquitetura foi pensada, que permite distribuir os benefícios da nuvem, tornando-a mais próxima do cliente final. A computação em névoa ou computação de borda, se caracteriza por permitir armazenamento, controle e comunicação próximos da aplicação (8).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Testar *fog computing* como uma alternativa de arquitetura para internet das coisas, visando entender os benefícios e os problemas dessa arquitetura em comparação com a *cloud*, apresentando uma noção geral do desempenho do ambiente em diferentes situações.

1.1.2 Objetivos Específicos

- Testar o funcionamento dos sistemas da *Amazon Web Services* como soluções para criar um ecossistema de internet das coisas.
- Analisar os recursos do *Amazon FreeRTOS* na placa *ESP32* interagindo com as soluções da *Amazon Web Services*.
- Obter e analisar medidas em relação a latência, escalabilidade.
- Analisar soluções para tomada de decisão em caso de interrupção ou falha de dispositivos na rede.

1.2 JUSTIFICATIVA

A tendência de computação em nuvem está encontrando crescentes desafios em diversos novos requisitos da internet das coisas (*IOT*), como latência, escalabilidade, largura de banda, serviços ininterruptos com conectividade intermitente com a nuvem, além dos desafios de segurança, como autenticação de dispositivos, tomada de decisão em casos de falhas e atualização de serviços ininterruptos. Uma solução que tem sido adotada para resolver esses desafios é a *fog computing*.

Para garantir a segurança do ecossistema *IOT* são necessários testes para averiguar que os problemas encontrados na *cloud* estão sendo sanados com a utilização de *fog*, dessa forma garantindo que a existência de novos desafios ou deficiências no ecossistema *IOT* sejam minimizados. É necessário investigar as métricas apresentadas pela *fog* quando submetidas a testes que buscam resolver os desafios empregados pela *cloud*, ou encontrar parâmetros que demonstram casos onde a *fog* é mais viável que a *cloud*, apresentando também como ambas são mutuamente benéficas. Desse modo, podemos fazer uma comparação confiável entre essas duas arquiteturas e como utilizá-las em paralelo.

2 REFERENCIAL TEÓRICO

2.1 INTERNET DAS COISAS (*IOT*)

O termo internet das coisas foi usada primeiramente por Kevin Ashton para descrever sistemas onde objetos podiam se comunicar via sensores, dependendo ou não da interação humana. O conceito se baseia em tornar os computadores inteligentes para que possam entender e processar os dados que até então são apenas humanamente interpretados. “Nós poderíamos rastrear e contar tudo, reduzir muito o gasto, perda e custo”[1] (3). Segundo Ashton (3), a internet das coisas permite que os computadores, através do uso de sensores, tenham sentidos humanos, ou seja, seriam capaz de ver, ouvir e detectar odores.

Segundo Atzori, Iera e Morabito (5), a internet das coisas é um paradigma que cresce rapidamente e tem um grande impacto no dia a dia dos usuários, com diversas aplicações como domótica, assistentes pessoais, e *e-health*, além dos impactos na indústria, com automação, manufatura, gerenciamento de processos e logística. Entretanto ainda há várias questões a serem tratadas para que a internet das coisas seja totalmente aceita e possa estar mais presente na nossa vida. Os problemas centrais envolvem interoperabilidade, confiabilidade, privacidade e segurança (5).

A figura1 representa um ecossistema *IOT*. Na figura é possível reparar a presença da nuvem, que é onde todos os dispositivos estão ligados e por ela interagem uns com os outros.

Figura 1 – Ilustração de um ecossistema *IOT*

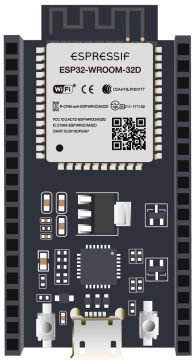
Fonte: Pixabay (15)

[1] Tradução nossa

2.2 ESP32

Esta sessão apresenta algumas informações do hardware escolhido para as análises do projeto, a placa microcontroladora *ESP32*. Todas as informações referentes a placa foram retiradas do *datasheet* da fabricante da placa. A placa *ESP32* é uma placa de baixo consumo energético e foi projetada para aplicações móveis, vestíveis (*wearables*) e de internet das coisas. O módulo *ESP32* conta com 11 portas *GPIO* com funções de *PWM*, *I2C*, *SPI* entre outros, ainda possui *Wi-Fi*, *Bluetooth*, antena embutida, conector micro-usb e opera entre 4,5v e 9v. A figura2 apresenta um *ESP32 DevKitC*.

Figura 2 – Ilustração de um *ESP32DevKitC*



Fonte: Espressif (10)

A tabela1 mostra as principais especificações da placa (33).

Tabela 1 – Especificações da placa ESP32

CPU	Xtensa Dual-Core 32-bit LX6
ROM	448 KBytes
RAM	520 KBytes
Flash	4 MB
Clock máximo	240 MHz

2.3 RASPBERRY PI

A placa *Raspberry Pi 3 model B* foi o dispositivo escolhido para funcionar como núcleo do ecossistema *fog*.

Um *Raspberry Pi* é um computador de tamanho reduzido, que possui conexões padrões para permitir interação com o usuário e permite a execução de um sistema operacional a partir de um cartão de memória *microSD*. O *Raspberry Pi 3 model B* que foi escolhido para este projeto possui um *SoC BCM2837A0/B0*, um *clock* de 1200MHz, possui 1 GB de memória RAM, 4 portas USB e *Wireless* (11).

O sistema operacional recomendado para a placa *raspberry pi* é o *Raspbian*. O *raspbian* é um sistema operacional de código livre baseado na distribuição *GNU/Linux Debian* que foi otimizada para ser executada no *raspberry pi*. O *Raspbian* possui mais de 35.000 pacotes de software pré-compilados para serem facilmente instalados (16). A figura3 apresenta um Raspberry Pi 3 Model B.

Figura 3 – Raspberry Pi 3 Model B

Fonte: Pi (14)

2.4 FREERTOS

Um sistema operacional de tempo real provê suporte básico para escalonamento, gerenciamento de recursos, sincronização, comunicação, cronometragem precisa e entrada e saída. Sistemas operacionais de tempo real enfatizam a previsibilidade, eficiência e incluem funcionalidades para suportar a redução do volume de tempo (32).

Aplicações de tempo real podem ser dividias em *soft real time* e *hard real time*. *Soft real time* são requisitos que estabelecem um prazo final, mas caso esse prazo final não for atendido, o sistema não ficará inutilizável. *Hard real time* são requisitos que estabelecem um prazo final e caso esse prazo final não seja cumprido, ocorrerá uma falha completa do sistema (6).

O *FreeRTOS* é um *kernel* de tempo real sobre o qual aplicações *hard real time* podem ser desenvolvidas. Ele permite que as aplicações sejam organizadas como coleções de independentes tarefas de execução. Cada tarefa é processada durante um tempo igual, sendo que a prioridade da tarefa define qual será executada primeiro, consequentemente, sendo as de maior prioridade executadas mais vezes (6).

2.5 CLOUD COMPUTING

Mell e Grance (13) definem computação em nuvem como sendo “um modelo para permitir acesso a rede onipresente conveniente e sob demanda a um conjunto compartilhado de

recursos de computação configuráveis" [2] (13).

O modelo de *cloud computing* é composto por cinco características essenciais: (13)

- Auto atendimento sob demanda: O consumidor pode provisionar recursos sem necessidade de interação humana com os provedores.
- Amplo acesso a rede: Os recursos disponíveis na rede são acessado por dispositivos padrões que promovem o uso por plataformas heterogêneas.
- Agrupamento de recursos: Os recursos são agrupados para atender múltiplos consumidores, que possuem diferentes recursos. Sem necessariamente o cliente ter conhecimento de onde esse recurso está sendo provido.
- Elasticidade: Os recursos podem ser escalados e liberados de acordo com a necessidade do usuário.
- Medição de serviço: Recursos podem ser controlados, monitorados e relatados para o usuário e para o provedor garantindo transparência.

Quatro modelos de implantação: (13)

- *Private cloud*: A infraestrutura é provida exclusivamente para uma organização, normalmente gerenciada e operada pela organização.
- *Community cloud*: A infraestrutura é provida para uma comunidade, normalmente gerenciada e operada por mais de uma organização.
- *Public cloud*: A infraestrutura é provida para o publico geral, normalmente é gerenciada e operada por organizações governamentais ou não governamentais, academia, ou uma combinação destes órgãos.
- *Hybrid cloud*: É a combinação de duas ou mais infraestrutura de *clouds* distintas

Serviços *cloud* estão se tornando uma arquitetura poderosa para realizar tarefas complexas em larga escala, além de abranger uma gama de funções de TI que vão desde armazenamento e computação, para serviços de banco de dados e aplicativos. A crescente necessidade de armazenamento, processamento e análise de uma grande quantidade de dados levou muitas organizações e usuários a adotar *cloud compunting* (12).

Um dos fatores que favorecem o uso de *cloud computing* é a internet das coisas. A internet das coisas tem se tornado cada dia mais presente. A grande quantidade de dados gerados por dispositivos que compõem o ecossistema de internet das coisas e a necessidade de armazenar e processar esses dados tornam o uso de *cloud computing* uma solução provável (1).

[2] Tradução nossa

2.6 FOG COMPUTING

Fog computing estende a *cloud computing* para a borda da rede, provendo computação, armazenamento, e serviços de rede. Ela habilita uma gama de aplicações que encontram desafios quando trabalham apenas com a *cloud*, há uma interação frutífera envolvendo *cloud* e *fog computing*, onde a *fog* existe entre os dispositivos finais e os data centers de *cloud computing* (8).

Fog computing é caracterizada por ter baixa latência, distribuição geográfica generalizada, mobilidade, grande número de nós, presença forte de transmissão e aplicações de tempo real (8).

A necessidade da criação dessa nova arquitetura se dá pelas dificuldades que algumas aplicações tem de centralizar seus serviços na *cloud*. Uma onda emergente de implantações de internet, mais notadamente a internet das coisas, requer suporte a mobilidade e distribuição geográfica além de localização consciente e baixa latência (9).

Atualmente *cloud computing* não é pensada para o volume, variedade e velocidade de dados que a internet das coisas gera. Os principais desafios encontrados na utilização de *cloud* para esse tipo de aplicação são citados abaixo (9).

- Latência: Muitas aplicações precisam garantir que em determinadas situações, o tempo de resposta ou de atuação ocorra em poucos milissegundos. Como por exemplo, sistemas industriais onde após a leitura de um determinado sensor, um atuador aja em resposta imediatamente. Ainda é possível citar aplicações de tempo real que necessitam de previsibilidade em suas respostas. Em ambos os casos, a interação com a *cloud*, não garante uma latência necessária para suprir estas necessidades.

- Largura de banda: O crescimento da quantidade de dispositivos conectados tem gerado um gargalo gigante de dados. Uma única aplicação pode gerar muitos dados a partir de seus sensores que interagem com outras máquinas ou informações que são geradas para o usuário. Enviar essa grande quantia de dados para a *cloud* iria requerer uma largura de banda muito elevada, além de ser muitas vezes desnecessário mandar toda essa quantidade de dados, entra em risco a privacidade do usuário.

- Serviços ininterruptos: É muito difícil garantir com a *cloud* a ininterrupção de sistemas que necessitam estar constantemente conectada a ela. Nos exemplos que remetem a esse caso, os desastres causados pela perda de conexão podem ser gravíssimos.

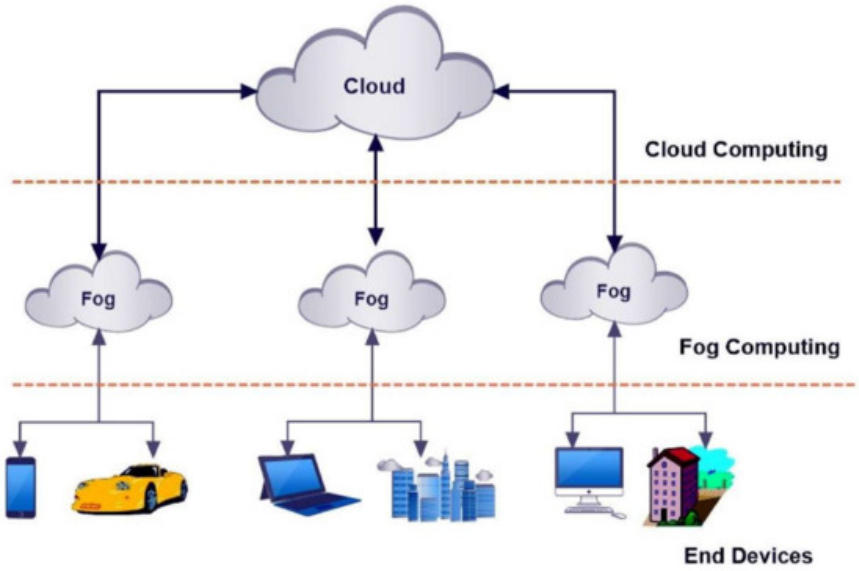
- Segurança: Existem vários desafios de segurança no modelo de conexão com a *cloud*, como por exemplo:

  - Mandar a credencialidade de um grande números de dispositivos: Se torna impraticável conectar tantos dispositivos com a *cloud* para realizar atualizações de software e garantir a credencialidade do dispositivo na rede

– Proteção de dispositivos com recursos limitados: É necessário garantir que nenhum componente da rede seja faltoso.

A figura4 demonstra a conexão entre dispositivos com a *fog* e a integração entre várias *fogs* com a *cloud*

Figura 4 – Ilustração de conexão com *fog* e *cloud*



Fonte: Atlam, Walters e Wills (4)

2.7 AMAZON WEB SERVICES

Nesta sessão são apresentados os serviços fornecidos pela *Amazon Web Services (AWS)*. Todas as informações apresentadas, foram retiradas do site da *AWS*.

Os produtos para soluções *IOT* da *AWS* permitem criar um ecossistema *IOT* inteiro, desde a borda até a nuvem. São fornecidos softwares de dispositivo, serviços de controle e serviços de dados. A integração entre todos os sistemas da *AWS* se tornam muito simples.

2.7.1 AWS IoT Greengrass

O *AWS IoT Greengrass* é um software que estende os recursos da nuvem para dispositivos locais, as informações são coletadas e analisadas mais próximas dos dispositivos que geraram estas informações permitindo que estes dispositivos reajam aos eventos de forma autônoma e se comuniquem entre si com segurança. O software gerencia as aplicações baseado em nuvem, implementando localmente funções lambda e conectores que podem ser acionados por eventos locais, mensagens na nuvem ou outras origens (29).

A troca de mensagens é garantida sem precisar de conexão com a nuvem, fornecendo um *buffer* para que as mensagens que entram e saem da nuvem possam ser preservadas em caso de falta de conexão (29).

O usuário do *AWS IoT Greengrass* está protegido por meio de autenticação e autorização dos dispositivos, por meio de uma conectividade segura na rede local e entre dispositivos locais e a nuvem. As credenciais de um grupo funcionam localmente até serem revogadas, em caso de interrupção na conectividade com a nuvem, os dispositivos podem se comunicar localmente (29).

Um grupo *AWS IoT Greengrass* é uma coleção de configurações e componentes. São usados para definir um escopo de interações. Existem dois tipos de dispositivos definidos no grupo, o núcleo e o dispositivo conectado ao núcleo (29).

O núcleo é o dispositivo que executa o software *AWS IoT Greengrass* e se comunica diretamente com o *AWS IoT* e o *AWS IoT Cloud Services*. Um núcleo possui seu próprio certificado para se autenticar com o *AWS IoT*. O núcleo executa *Lambda* local, um agente de implantação e um rastreador de endereços IP que envia informações ao *AWS IoT Greengrass Cloud Services* para permitir que dispositivos *Greengrass* se conectem ao núcleo. Existe apenas um dispositivo núcleo em cada grupo (29).

Os dispositivos conectados ou dispositivos *Greengrass* se conectam a um núcleo do mesmo grupo *Greengrass*. Os dispositivos podem executar Amazon *FreeRTOS*, utilizar *AWS IoT Device SDK* ou *AWS IoT Greengrass Discovery API* para obter as informações de conexão com o núcleo. Cada dispositivo *Greengrass* possui seu próprio certificado, seu próprio *shadow device* e registro do dispositivo no *AWS IoT* (29).

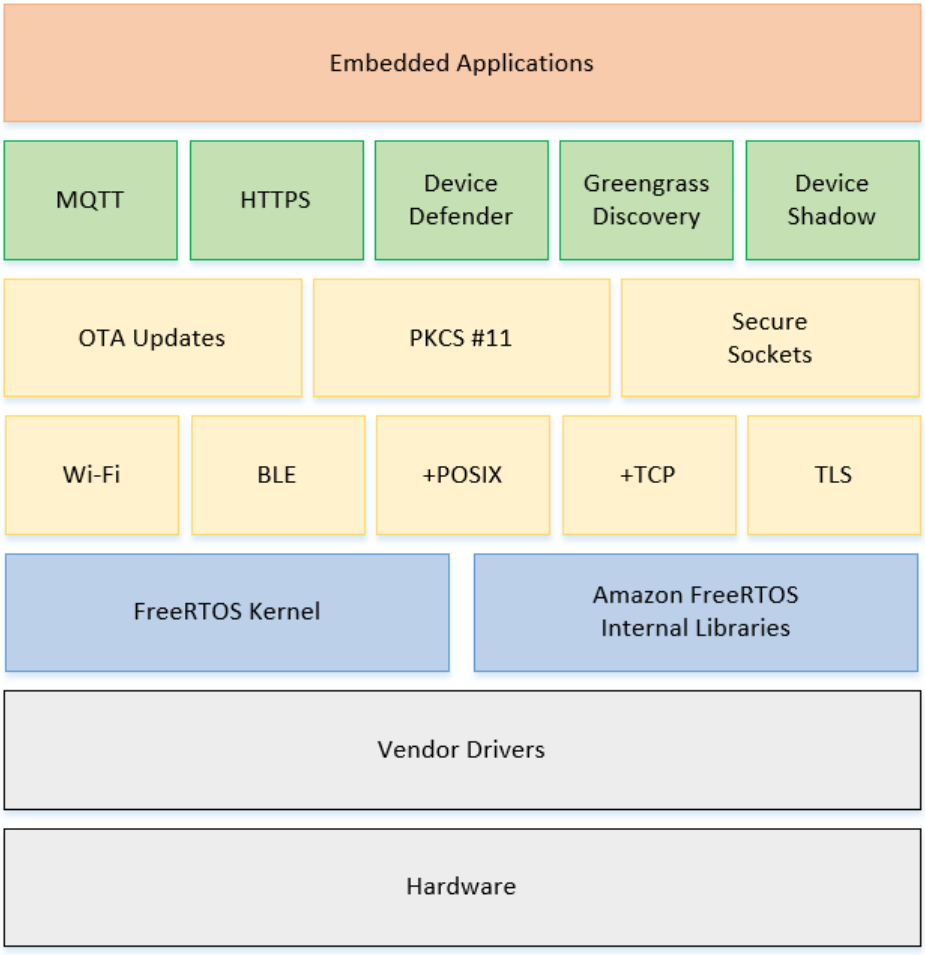
Em um grupo *Greengrass* é possível criar assinaturas que permitem comunicação de dispositivos com funções lambda, conectores, outros dispositivos, *AWS IoT* ou outro serviço *shadow* via o protocolo *MQTT*, as mensagens são roteadas por meio do núcleo (29).

2.7.2 Amazon FreeRTOS

O Amazon *FreeRTOS* é um projeto de código aberto que expande o sistema operacional de tempo real *FreeRTOS*, adicionando bibliotecas para conectividade, segurança e atualizações remotas. As bibliotecas permitem conexão segura dos dispositivos com o *AWS IoT* via *MQTT* e sombras do dispositivo, descoberta e conexão com um núcleo *AWS IoT Greengrass* (28).

A imagem compilada do Amazon *FreeRTOS* possui a funcionalidade dos aplicativos criadas pelo desenvolvedor, as bibliotecas de softwares fornecidas pela Amazon, o kernel *FreeRTOS* e os pacotes de suporte a drivers e placas (28). A figura5 apresenta a arquitetura do Amazon *FreeRTOS*.

Figura 5 – Arquitetura Amazon FreeRTOS



Fonte: Services (28)

2.7.3 AWS Lambda

A *AWS Lambda* é um serviço que permite execução de código sem provisionamento ou gerenciamento de servidor. O código é executado somente quando necessário, fazendo toda a administração dos recursos computacionais, manutenção de servidor e sistema operacional, provisionamento de capacidade e escalabilidade de forma automática além de monitoramento do código e registro em log. O código pode ser executado em resposta a eventos (18).

2.7.4 AWS IoT

O *AWS IoT* permite comunicação bidirecional segura entre dispositivos conectados à internet e a nuvem da Amazon, permitindo assim a coleta de dados de telemetria de vários dispositivos, armazenagem e análise desses dados. Entre os componentes que o *AWS IoT* possui, estão (17):

- Operador de mensagens: Fornece um mecanismo seguro para comunicação entre dispositivos e aplicativos *AWS IoT*, permitindo publicação e recepção de mensagens. Permitindo a utilização do protocolo *MQTT* para publicar e se inscrever e do protocolo *REST HTTP* para publicar mensagens (17).
- Registro: Componente para organizar recursos de dispositivos na nuvem *AWS*, sendo possível também associar certificados e IDs de cliente *MQTT* a cada dispositivo (17).
- Registro de grupo: Permite gerenciar vários dispositivos ao mesmo tempo. Pode haver uma hierarquia de grupos e todas as permissões concedidas a um grupo são aplicadas a todos os dispositivos e grupos filhos integrantes do grupo (17).
- *Device shadow*: Documento usado para recuperar o estado atual de um dispositivo (17).
- Serviço *device shadow*: Serviço que representa persistentemente o seu dispositivo na nuvem, é possível publicar informações em um *device shadow* e o dispositivo sincronizará as informações quando conectado (17).
- Serviço de provisionamento de dispositivos: Utiliza um modelo descrevendo os recursos necessários de um dispositivo para provisioná-lo. Esse modelo pode ser uma coisa, um certificado ou uma política. Uma coisa é uma entrada no registro contendo atributos que descrevem um dispositivo. O certificado permite autenticação dos dispositivos com a *AWS IoT*. As políticas determinam as operações que o dispositivo pode executar na *AWS IoT* (17).

2.8 TRABALHOS RELACIONADOS

Para realizar a revisão bibliográfica, foram utilizados os mecanismos de busca *scholar google, IEEE, ACM*, documentação disponível pela *AWS, Espressif, FreeRTOS* e repositório de universidades. Nenhum dos trabalhos encontrados analisava os serviços da *AWS* como solução para criar um ecossistema *IOT* utilizando *fog computing*. Os trabalhos relacionados que basearam essa monografia estão listados a seguir:

- O trabalho da Bonetti (7) é um estudo de caso para analisar desempenho de *fog computing* para internet das coisas utilizando a plataforma *Kaa IOT*. Os testes efetuados são para enviar e armazenar amostras de temperatura e para enviar e processar as amostras de temperatura em um ambiente *cloud e fog*, analisando a latência média.
- Chiang e Zhang (9) demonstram exemplos de desafios que ocorrem ao trabalhar com a *cloud computing* e como a *fog computing* pode resolver tais desafios.
- O artigo de Hashem et al. (12) discute a relação entre *big data* e *cloud computing*. Este trabalho permite entender melhor alguns desafios encontrados por conta da grande quantidade de dados que existe trafegando na rede nos dias atuais.

3 ESTUDO DE CASO

3.1 METODOLOGIA

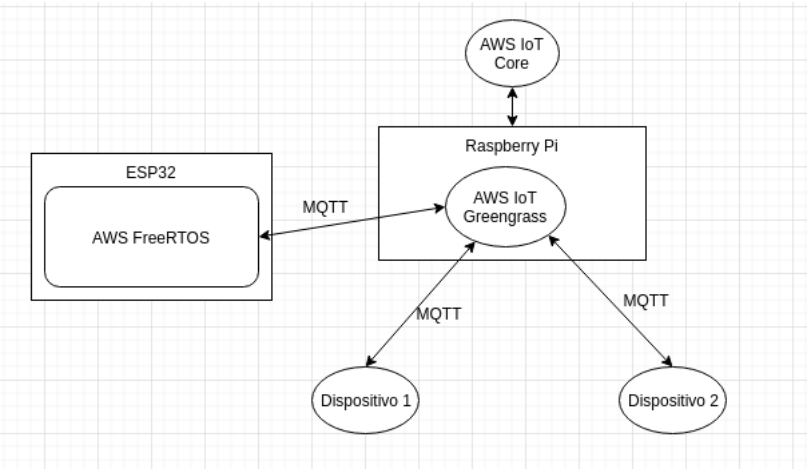
A metodologia proposta para este estudo de caso é a criação de um ecossistema *IOT*, usando os serviços *AWS* para simular o ambiente utilizando *fog computing*. Os testes têm como objetivo avaliar como a *fog computing* resolve as brechas que existem nesse tipo de aplicação, quando usado apenas *cloud computing*, sem a existência de um *middleware*. Além disso, será avaliado a integração dos sistemas *AWS*.

Para criar o ecossistema *IOT*, foi usado a placa *Raspberry pi 3 model B* com o sistema operacional *Raspbian* executando o serviço *Amazon Greengrass* e a placa *ESP32* e embarcada uma imagem *Amazon FreeRTOS* executando a aplicação de teste. A placa se comunica com o dispositivo de borda *Raspberry*.

3.2 DISPOSIÇÃO DOS COMPONENTES

O esquema do ecossistema construído para o estudo de caso é apresentado na figura 6.

Figura 6 – Fluxograma da disposição do ecossistema



Fonte: Do Autor

3.3 CONFIGURAÇÃO DOS CENÁRIOS AVALIADOS

Nesta seção são apresentados o desenvolvimento das configurações dos softwares da *AWS* nas placas *Raspberry pi* e *ESP32*.

3.3.1 Raspberry pi e AWS IoT Greengrass

Para realização dos testes o dispositivo núcleo escolhido foi o *Raspberry pi 3 Model B*, este dispositivo é responsável por executar o software *AWS IoT Greengrass Core*.

A versão do sistema operacional no *raspberry pi* é o *Raspbian Buster*. Para execução do *AWS IoT Greengrass* foi necessário o *download* de dependências e configurações de usuário e grupo no *raspbian*. A configuração de usuário e verificação de dependência seguida está disponível em(22). As informações sobre download das chaves geradas e da execução do *AWS IoT Greengrass* no *Raspberry Pi* está disponível em(23).

3.3.1.1 Configurações AWS IoT

Antes de iniciar o *AWS IoT Greengrass* core no *Raspberry Pi*, foi necessário a criação de um grupo, que pode ser feita a partir do console de gerenciamento da *AWS*. Ao criar o grupo é necessário permitir a criação de uma política de regra de acesso aos recursos *AWS IoT Greengrass* (24).

Terminado essas configurações básicas, o *AWS IoT* cria um grupo *AWS IoT Greengrass* com as políticas de segurança escolhidas e prepara os arquivos de configuração para serem instalados no dispositivo (24).

3.3.1.2 Instalação

Após o download do software *AWS IoT Greengrass* com as configurações do grupo criadas pelo console *AWS IOT*, foi necessário descompactar os arquivos para um diretório local no *Raspberry Pi*.

O *AWS IoT Greengrass Core* consiste em quatro diretórios, o diretório *certs* onde ficam armazenados todos os certificados e chaves, o diretório *config* onde se localizam os arquivos de configurações, o diretório *ggc* no qual ficam armazenados os arquivos binários do *Greengrass Core* e o diretório *ota* responsável pelo agente de atualização *Over-The-Air* que é o modo mais indicado para fazer atualizações no dispositivo núcleo.

Para continuar com a configuração, foi necessário inserir certificados no diretório *certs*. Os certificados fornecem as permissões de acesso aos recursos *AWS IoT*. Foi usado o *Amazon Trust Service (ATS) endpoints* e os certificados *ATS root CA*, com esses certificados a comunicação com o *AWS IOT* utilizando o protocolo *MQTT* foi habilitada (23).

Para executar o *Greengrass* basta executar o arquivo *greengrassd* disponível no diretório *ggc/core*. Após a execução do arquivo binário, o serviço executa em *daemon* e o *PID* é apresentado (23).

3.3.2 ESP32 e Amazon FreeRTOS

Para usar o *Amazon FreeRTOS* é necessário inicialmente configurar a conta e as permissões da *AWS*. É necessário adicionar um usuário *IAM* a conta *AWS* e anexar as seguintes políticas do *IAM* a conta de usuário *IAM: AmazonFreeRTOSFullAccess, AWSIoTFullAccess* (21).

A segunda etapa de configuração necessária é o registro da placa *MCU* com o *AWS IoT*. Para registrar a placa com a *AWS IoT*, são necessárias as seguintes configurações (30):

- Uma política *AWS IoT*: Necessária para conceder permissões ao dispositivo para acessar recursos *AWS IoT*
- Uma coisa no *AWS IoT*: A coisa é de fato o dispositivo apresentado para a *AWS IoT*, a partir da coisa a *AWS IoT* pode gerenciar o dispositivo.
- Uma chave privada e um certificado *X.509*: Garantem autenticação com o *AWS IoT*

A política no *AWS IoT* pode ser criada pelo próprio console do *AWS IoT*, a política padrão configurada concede as seguintes permissões (30):

- *iot:Connect*: Garante ao dispositivo conexão com o agente de mensagens da *AWS IoT*
- *iot:Publish*: Permite ao dispositivo publicar mensagens *MQTT* em qualquer tópico *MQTT*.
- *iot:Subscribe*: Permite ao dispositivo assinar o filtro de tópicos *MQTT*
- *iot:Receive*: Garante ao dispositivo permissão para receber mensagens do agente de mensagens em qualquer tópico *MQTT*.

Após a criação da política, é necessário no console do *AWS IoT*, criar a coisa, adicionar o dispositivo ao registro de coisas, adicionar um certificado a essa coisa, baixar a chave privada e o certificado da coisa, ativar o certificado, anexar a política que concede ao dispositivo acesso a operações do *AWS IoT* e então registrar a coisa (30).

Algumas placas como o *ESP32* não necessitam que o registro da coisa seja feito no console da *AWS IoT*, as configurações são feitas diretamente nos arquivos que compõem o Amazon *FreeRTOS* e quando executado o *script* de *setup*, a coisa é registrada automaticamente (26).

Após o termino dessas configurações iniciais, o download do Amazon FreeRTOS pode ser feito [1].

Para a placa *ESP32* é necessário a instalação do sistema de compilação *CMake*, que é o sistema responsável por fazer compilação cruzada, e o *drivers* necessários para garantir a comunicação serial com a placa *ESP32* (26).

Após concluir as configurações, foi possível gerar um *build* utilizando o *cmake* e então gravar o *build* na memória flash da placa *ESP32* (26).

[1] O download do Amazon FreeRTOS está disponível em https://github.com/aws/amazon-freertos.

Existe uma série de bibliotecas disponíveis pela Amazon para o Amazon *FreeRTOS* que permitem a comunicação do sistema operacional com os serviços *AWS*.

Neste trabalho, as bibliotecas que foram testadas para demonstração de um ecossistema *fog* são as bibliotecas *AWS IoT Greengrass* e *MQTT*, ambas as bibliotecas além de outras estão disponíveis no projeto Amazon *FreeRTOS*.

3.3.3 Biblioteca AWS IoT Greengrass

A biblioteca *AWS IoT Greengrass* é necessária para permitir ao dispositivo microcontrolador descobrir o dispositivo núcleo a partir das *APIs* de descoberta do *AWS IoT Greengrass*, após a descoberta do núcleo, é possível trocar mensagens (20).

O uso desta biblioteca requer apenas a criação de um item na *AWS IoT* e nela a inclusão de um certificado e uma política. Para permitir a descoberta do núcleo pelo dispositivo alguns arquivos de configuração precisam ser editados. No arquivo *aws_clientcredential.h* os seguintes parâmetros precisam ser alterados (20):

- *clientcredentialMQTT_BROKER_ENDPOINT*: O endpoint da AWS IoT.
- *clientcredentialIOT_THING_NAME*: O nome da coisa no AWS IoT.
- *clientcredentialWIFI_SSID*: SSID da rede Wi-Fi.
- *clientcredentialWIFI_PASSWORD*: Senha da rede Wi-Fi.
- *clientcredentialWIFI_SECURITY*: Tipo de segurança utilizado pela rede Wi-Fi.

No arquivo *aws_clientcredential_keys.h* é inserido o certificado *PEM* e a chave privada *PEM* associada a coisa (20).

Após alterado esses parâmetros, é possível encontrar o núcleo *Greengrass*.

3.3.4 Biblioteca Amazon FreeRTOS MQTT

A biblioteca de *MQTT* disponível permite a publicação e assinaturas de tópicos *MQTT*. Essa biblioteca é essencial devido a importância do protocolo *MQTT* para comunicação com o *AWS IoT* e o *Greengrass* (19).

A biblioteca possui uma *API* para comunicação totalmente assíncrona, é paralelizável para altas taxas te transferência e apresenta desempenho escalável (19).

3.3.5 Shadow Devices

Os *shadow devices* ou dispositivos sombra, são uma ótima maneira de conseguir se comunicar com um dispositivo que está momentaneamente indisponível na rede. O *shadow*

device é um arquivo em formato *JSON* usado para recuperar informações sobre o estado atual de um *dispositvo*, caso esse dispositivo esteja indisponível a comunicação com seu dispositivo sombra pode continuar ocorrendo e quando o dispositivo se conectar novamente a rede, ele poderá fazer a sincronização com seu dispositivo sombra (31).

Para realização dos testes de *shadow devices*, foram criados dois dispositivos, um dispositivo ficava responsável por escrever no *shadow device* e o segundo ficava responsável por atualizar seu estado a partir do *shadow device* (27).

Para a realização desse teste, era necessário que cada dispositivo fizesse mais de uma assinatura aos tópicos de mensagem. Os tópicos assinados e suas funções podem ser verificados na Tabela 2 (25).

Tabela 2 – Tópicos shadow device

Tópico	Função
aws/things/thingName/shadow/update	Atualizar o tópico
aws/things/thingName/shadow/update/accept	Confirma a atualização
aws/things/thingName/shadow/update/rejected	Rejeita a atualização
aws/things/thingName/shadow/update/delta	Envia atualização recebida

Fonte: (25)

3.4 AVALIAÇÃO DE LATÊNCIA

A latência é a medida de tempo entre o envio completo de um pacote da origem até um destino (2). A partir da medida da latência, podemos avaliar performance do ecossistema *fog*, tendo em vista a importância da baixa taxa de latência nos sistemas de *IoT*, em comparação com a *cloud*.

Para avaliação dos testes de latência, foi criado dois dispositivos pertencentes ao grupo do *Greengrass*, no qual o *Raspberry Pi* desempenhava a função de núcleo.

O dispositivo que iniciava a comunicação ficava responsável por escrever mensagens *MQTT* em um tópico e começar a contar o tempo, também esperava receber mensagens em um outro tópico para parar de contar o tempo. O segundo dispositivo apenas esperava as mensagens chegar no tópico em que estava inscrito, ao receber um mensagem, apenas devolvia essa mensagem num segundo tópico.

Para os testes de latência com a *cloud*, foi criada uma função *lambda* no console *AWS IoT* que executava toda vez que um determinado tópico recebia dados, retornando a mensagem para um segundo tópico. Um dispositivo devidamente cadastrado no console *AWS IoT* enviava mensagens para o primeiro tópico, começava a contar o tempo e ficava esperando a resposta no segundo tópico. Quando recebia a mensagem de retorno, calculava o tempo passado entre o envio e a recepção da mensagem.

Os testes foram efetuados em ambientes controlados, os únicos dispositivos conectados ao roteador eram o *Raspberry Pi* e os dois dispositivos responsáveis por se comunicar entre si durante o teste de latência para *fog*. Durante o teste de latência para *cloud*, apenas o *Raspberry Pi* e um dispositivo ficou conectado ao roteador, como o serviço *AWS IoT Greengrass* ainda não está disponível na América do Sul, os testes de latência com a *cloud* ocorreram utilizando a *cloud* da região oeste dos Estados Unidos da América. O link utilizado possuía uma banda de 80 Mb.

4 RESULTADOS E ANÁLISE

Neste capitulo, serão apresentados os resultados obtidos a partir dos testes de latência e das análises feitas a partir dos testes de comunicação utilizando *MQTT*, dos testes envolvendo as bibliotecas do *FreeRTOS* no *ESP32* e os testes com os *shadow devices*.

A partir da aplicação dos testes, dos resultados obtidos e dos recursos analisados, é possível discutir de maneira mais precisa a utilização da plataforma *AWS* como solução para *fog computing*.

4.1 LATÊNCIA

A tabela 3 apresenta a amostra recolhida da comunicação entre dois dispositivos conectados a *fog*. A latência pode ser observada para diferentes tamanhos de pacotes.

Tabela 3 – Amostra de Latência entre dois dispositivos se comunicando na *Fog*

32 B	64 B	128 B	256 B	512 B	1024 B	2048 B	4096 B
2,6ms	4,3ms	3,8ms	4,4ms	3,3ms	8,9ms	2,6ms	3,3ms
3,9ms	3,9ms	3,2ms	3,3ms	2,3ms	3,3ms	4,6ms	12,5ms
2,1ms	3,0ms	3,1ms	3,3ms	3,2ms	3,2ms	4,0ms	3,6ms
3,1ms	3,5ms	2,2ms	3,7ms	1,3ms	7,5ms	2,5ms	4,2ms
2,2ms	5,0ms	2,2ms	3,2ms	2,9ms	3,4ms	3,2ms	3,1ms
3,1ms	3,0ms	4,3ms	2,0ms	1,4ms	3,2ms	2,6ms	12,6ms
2,8ms	3,0ms	1,3ms	3,6ms	5,8ms	6,4ms	4,6ms	7,6ms
2,9ms	3,1ms	2,0ms	1,2ms	1,9ms	2,3ms	3,7ms	4,9ms
6,5ms	5,0ms	4,3ms	3,2ms	4,9ms	4,9ms	2,4ms	3,9ms
1,1ms	4,1ms	4,5ms	2,6ms	4,4ms	3,9ms	6,1ms	3,1ms
2,5ms	3,0ms	2,2ms	3,3ms	4,5ms	3,3ms	4,9ms	5,0ms
3,2ms	5,5ms	4,2ms	2,9ms	2,4ms	3,1ms	11,1ms	6,4ms
1,1ms	5,7ms	3,2ms	3,4ms	3,5ms	6,8ms	4,6ms	3,8ms
1,1ms	2,0ms	2,4ms	3,3ms	2,8ms	3,3ms	15,2ms	4,3ms
4,4ms	2,4ms	3,3ms	5,9ms	3,4ms	7,1ms	3,8ms	14,1ms
1,2ms	6,2ms	3,1ms	3,3ms	3,9ms	3,4ms	2,5ms	6,8ms
1,2ms	2,1ms	4,2ms	3,2ms	2,6ms	7,4ms	4,0ms	11,2ms
2,2ms	2,9ms	3,2ms	6,7ms	3,5ms	3,3ms	7,2ms	3,5ms
1,1ms	4,2ms	4,9ms	4,0ms	2,4ms	3,2ms	13,9ms	4,8ms
4,0ms	1,9ms	2,6ms	3,2ms	2,3ms	3,5ms	7,6ms	3,2ms
1,1ms	2,2ms	1,2ms	4,0ms	2,7ms	3,4ms	4,1ms	4,5ms
1,1ms	2,6ms	3,2ms	3,2ms	5,4ms	3,4ms	2,5ms	3,8ms
4,4ms	1,5ms	4,9ms	2,9ms	5,5ms	3,4ms	2,4ms	25,4ms
1,4ms	1,4ms	3,2ms	2,0ms	4,4ms	4,5ms	54,2ms	12,8ms
1,0ms	3,1ms	3,3ms	3,3ms	8,3ms	3,2ms	3,0ms	11,6ms
3,5ms	3,2ms	5,2ms	1,3ms	4,4ms	3,3ms	41,3ms	20,1ms
1,4ms	2,0ms	4,1ms	5,2ms	2,5ms	3,2ms	6,7ms	13,2ms
3,3ms	2,4ms	3,2ms	1,7ms	3,3ms	3,9ms	2,8ms	14,5ms
1,6ms	3,2ms	2,2ms	3,6ms	5,4ms	3,3ms	4,8ms	19,2ms
1,1ms	4,1ms	3,2ms	3,7ms	2,4ms	3,7ms	2,5ms	6,3ms

Fonte: Do Autor

A tabela 4 apresenta a amostra recolhida da comunicação entre dois dispositivos conectados a *cloud*. A latência pode ser observada para diferentes tamanhos de pacotes.

Tabela 4 – Amostra de Latência entre dois dispositivos se comunicando na *Cloud*

32 B	64 B	128 B	256 B	512 B	1024 B	2048 B	4096 B
248,1ms	249,4ms	254,7ms	279,7ms	257,5ms	287,7ms	280,7ms	299,3ms
248,0ms	249,7ms	245,3ms	268,7ms	265,6ms	297,5ms	289,2ms	281,4ms
251,5ms	249,1ms	252,2ms	254,2ms	272,5ms	296,1ms	280,5ms	289,5ms
250,1ms	261,0ms	249,6ms	270,1ms	272,3ms	329,5ms	297,8ms	292,9ms
248,8ms	251,3ms	245,6ms	268,1ms	264,4ms	292,4ms	290,4ms	272,7ms
245,2ms	251,1ms	255,6ms	257,4ms	281,7ms	291,9ms	297,3ms	294,4ms
249,2ms	249,1ms	249,7ms	269,5ms	260,2ms	291,6ms	283,5ms	413,9ms
246,7ms	248,4ms	251,4ms	273,0ms	262,2ms	283,1ms	277,9ms	284,7ms
248,4ms	250,5ms	251,6ms	264,7ms	284,4ms	278,3ms	320,4ms	295,1ms
246,5ms	249,3ms	249,4ms	254,8ms	270,0ms	269,2ms	296,5ms	289,9ms
249,3ms	251,8ms	250,0ms	266,1ms	265,0ms	281,5ms	281,8ms	289,6ms
258,8ms	257,7ms	299,9ms	255,5ms	272,2ms	288,6ms	280,4ms	295,2ms
247,6ms	248,6ms	254,2ms	257,1ms	272,5ms	292,6ms	289,4ms	293,5ms
246,9ms	249,9ms	263,9ms	275,9ms	258,8ms	282,9ms	301,1ms	293,7ms
249,1ms	249,8ms	259,2ms	256,3ms	262,8ms	314,4ms	294,5ms	284,3ms
254,7ms	250,1ms	253,1ms	276,3ms	274,3ms	295,7ms	280,5ms	289,0ms
247,1ms	251,5ms	250,7ms	263,5ms	277,3ms	273,2ms	297,8ms	321,1ms
254,2ms	252,3ms	250,4ms	263,4ms	270,0ms	280,7ms	283,5ms	291,1ms
247,7ms	246,8ms	249,9ms	271,8ms	280,7ms	265,7ms	269,9ms	297,0ms
246,2ms	256,1ms	246,5ms	263,0ms	277,8ms	280,1ms	278,1ms	271,1ms
248,5ms	250,4ms	250,2ms	287,3ms	268,2ms	279,2ms	296,9ms	277,9ms
249,2ms	254,7ms	250,2ms	253,1ms	285,9ms	276,4ms	293,6ms	295,5ms
248,7ms	257,4ms	253,5ms	258,0ms	284,5ms	287,6ms	292,1ms	339,4ms
246,2ms	245,1ms	246,3ms	255,5ms	271,1ms	289,9ms	289,0ms	283,7ms
250,4ms	247,3ms	246,9ms	274,2ms	309,7ms	280,3ms	272,5ms	292,4ms
248,5ms	252,4ms	251,3ms	270,4ms	267,3ms	293,6ms	293,2ms	284,7ms
248,6ms	256,3ms	267,8ms	267,7ms	281,9ms	284,9ms	300,1ms	284,8ms
249,8ms	259,9ms	259,5ms	269,2ms	321,6ms	273,1ms	293,0ms	279,1ms
248,3ms	252,6ms	288,4ms	280,5ms	272,1ms	275,6ms	298,7ms	295,2ms
284,4ms	269,9ms	251,6ms	262,9ms	293,6ms	309,8ms	300,2ms	285,2ms

Fonte: Do Autor

A tabela 5 apresenta a média, desvio padrão e intervalo de confiança para cada amostra de cada pacote enviado para a *fog*.

Tabela 5 – Média, desvio padrão e intervalo de confiança das amostras de latência na *Fog*

Quantidade Bytes	Média (ms)	Desvio Padrão (ms)	Intervalo de confiança (ms)
32 B	2,4ms	1,3ms	0,5ms
64 B	3,3ms	1,2ms	0,4ms
128 B	3,2ms	1,0ms	0,4ms
256 B	3,4ms	1,2ms	0,4ms
512 B	3,6ms	1,5ms	0,5ms
1024 B	4,2ms	1,7ms	0,6ms
2048 B	7,8ms	11,3ms	4,0ms
4096 B	8,4ms	5,8ms	2,1ms

Fonte: Do Autor

A tabela 6 apresenta a média, desvio padrão e intervalo de confiança para cada amostra de cada pacote enviado para a *cloud*.

Tabela 6 – Média, desvio padrão e intervalo de confiança das amostras de latência na *Cloud*

Quantidade Bytes	Média (ms)	Desvio Padrão (ms)	Intervalo de confiança (ms)
32 B	250,2ms	6,9ms	2,5ms
64 B	252,3ms	5,0ms	1,8ms
128 B	255,0ms	11,7ms	4,2ms
256 B	266,3ms	8,7ms	3,1ms
512 B	275,3ms	13,9ms	5,0ms
1024 B	287,4ms	13,2ms	4,7ms
2048 B	290,0ms	10,3ms	3,7ms
4096 B	295,2ms	25,4ms	9,1ms

Fonte: Do Autor

4.2 DISCUSSÃO

A partir das bibliotecas disponíveis no Amazon *FreeRTOS* e testadas nesse trabalho foi possível constatar que a complexidade de conectar, gerenciar, comunicar dispositivos de maneira segura em uma rede *fog* é reduzida significativamente.

O comportamento dos *shadow devices* foi analisado a parte pois o seu propósito garante que o ecossistema *fog* se torne independente de dispositivos específicos, permitindo a comunicação e sincronização com uma *shadow* quando o dispositivo perde a conexão com o restante da rede. A sincronização também pode ocorrer constantemente com a *cloud* e em caso de interrupção, sincronizar posteriormente com o *shadow device*.

Considerando os resultados obtidos dos testes de latência, é possível perceber que a *fog* possui uma vantagem muito maior que a *cloud*, levando em consideração a localização da *cloud*

o teste demonstra um caso muito crítico onde a *cloud* está muito afastada de seu cliente final. A *fog* apresenta um resultado muito satisfatório de latência, permitindo assim a rápida ação em caso de alguma necessidade apontada por algum dispositivo.

Com os recursos avaliados e os resultados obtidos, é possível perceber, com a *fog* a resolução de requisitos apresentados pela internet das coisas que não são sanados pela *cloud computing*. Essa análise demonstra a necessidade desse ecossistema como recurso a aplicações *IoT*, não excluindo a *cloud* de seu papel, mas sim, agregando facilidade, agilidade e segurança em determinados processos.

5 CONCLUSÃO

O rápido crescimento do paradigma da internet das coisas exigiu algumas mundanças nas tecnologias existentes até então. Para resolver alguns requisitos apresentados pela internet das coisas como conectividade, latência, largura de banda, escalabilidade de dispositivos e de dados além de resolver as limitações encontradas em *cloud computing* a *fog computing* foi projetada.

O presente trabalho possibilitou avaliar as vantagens a aplicações de um ecossistema *fog* utilizando os recursos disponibilizados pela *Amazon Web Services* como suporte ao estudo de caso. A partir das análises de disponibilidade feitas e pelos dados coletados, foi possível confirmar uma vantagem da estrutura do ecossistema *fog* em relação a desempenho, autonomia, disponibilidade e segurança dos dispositivos, além da viabilidade de integração entre *fog computing* e *cloud computing* mesmo que a necessidade de integração intermitente entre esses ecossistemas não exista.

Tendo em vista a grande quantidade de bibliotecas para o Amazon *FreeRTOS* disponíveis, a grande quantidade de placas que tem suporte ao Amazon *FreeRTOS*, alguns testes apresentados aqui podem ser replicados. Como sugestão de trabalhos futuros se propõem alguns testes que não foram tratados nesse trabalho e possuem uma importância muito grande para análise de recurso da *fog*, como por exemplo as atualizações *OTA* e integração com outros tipos de serviços, além da distribuição de núcleos e dispositivos.

REFERÊNCIAS

1 AAZAM, M. et al. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In: PROCEEDINGS of 2014 11th International Bhurban Conference on Applied Sciences Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014. [S.l.: s.n.], jan. 2014. p. 414–419. DOI: 10.1109/IBCAST.2014.6778179.

2 ALMES, Guy; KALIDINDI, Sunil; ZEKAUSKAS, Matthew. **A round-trip delay metric for IPPM**. [S.l.], 1999.

3 ASHTON, Kevin. That 'Internet of Things' Thing. RFID Journal, jun. 2009.

4 ATLAM, Hany F.; WALTERS, Robert J.; WILLS, Gary B. Fog Computing and the Internet of Things: A Review. **Big Data and Cognitive Computing**, v. 2, n. 2, 2018. ISSN 2504-2289. DOI: 10.3390/bdcc2020010. Disponível em: <https://www.mdpi.com/2504-2289/2/2/10>.

5 ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The Internet of Things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2010.05.010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>.

6 BARRY, Richard. **Mastering the FreeRTOS Real Time Kernel**: A Hands-On Tutorial Guide. 161204. ed. [S.l.]: Real Time Engineers LTDA, 2016. 399 p. Disponível em: <https://www.freertos.org/Documentation/RTOS_book.html>.

7 BONETTI, Fernanda Da Silva. **ANÁLISE DE DESEMPENHO DE UMA ARQUITETURA DE FOG COMPUTING PARA INTERNET OF THINGS ATRAVÉS DE ESTUDOS DE CASO**. 2018. Monografia (Graduação) – Universidade Federal da Fronteira Sul, Chapecó, SC, Brasil.

8 BONOMI, Flavio et al. Fog Computing and Its Role in the Internet of Things. In: PROCEEDINGS of the First Edition of the MCC Workshop on Mobile Cloud Computing. Helsinki, Finland: ACM, 2012. (MCC '12), p. 13–16. ISBN 978-1-4503-1519-7. DOI: 10.1145/2342509.2342513. Disponível em: <http://doi.acm.org/10.1145/2342509.2342513>.

9 CHIANG, M.; ZHANG, T. Fog and IoT: An Overview of Research Opportunities. **IEEE Internet of Things Journal**, v. 3, n. 6, p. 854–864, dez. 2016. ISSN 2327-4662. DOI: 10.1109/JIOT.2016.2584538.

10 ESPRESSIF. **ESP32-DevKitC**. Disponível em: <https://www.espressif.com/en/products/hardware/development-boards>.

11 FOUNDATION, Raspberry Pi. **Raspberry Pi Documentation**. Disponível em: <https://www.raspberrypi.org/documentation/faqs/#introduction>. Acesso em: 23 nov. 2019.

12 HASHEM, Ibrahim Abaker Targio et al. The rise of "big data" on cloud computing: Review and open research issues. **Information Systems**, v. 47, p. 98–115, 2015. ISSN 0306-4379. DOI: https://doi.org/10.1016/j.is.2014.07.006. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0306437914001288>.

13 MELL, Peter M.; GRANCE, Timothy. **SP 800-145. The NIST Definition of Cloud Computing**. Gaithersburg, MD, United States, 2011.

14 PI, Raspberry. **Raspberry Pi 3 Model B**. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b>.

15 PIXABAY. **Rede Iot Internet Das Coisas**. Disponível em: <https://pixabay.com/pt/vectors/rede-iot-internet-das-coisas-782707/>. Acesso em: 25 jun. 2019.

16 RASPBIAN. **Welcome to raspbian**. Disponível em: <http://www.raspbian.org/>.

17 SERVICES, Amazon Web. **AWS IoT Core**. [S.l.: s.n.]. Disponível em: <https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/what-is-aws-iot.html>. Acesso em: 25 jun. 2019.

18 _____. **AWS Lambda**. [S.l.: s.n.]. Disponível em: <https://docs.aws.amazon.com/pt_br/lambda/latest/dg/welcome.html>. Acesso em: 25 jun. 2019.

19 _____. **Biblioteca Amazon FreeRTOS MQTT, versão 2.0.0**. Disponível em: <https://docs.aws.amazon.com/pt_br/freertos/latest/userguide/freertos-mqtt-2.html>.

20 _____. **Biblioteca de descoberta do AWS IoT Greengrass do Amazon FreeRTOS**. Disponível em: <https://docs.aws.amazon.com/pt_br/freertos/latest/userguide/freertos-lib-gg-connectivity.html>.

21 _____. **Configurar as permissões e a conta da AWS**. Disponível em: <https://docs.aws.amazon.com/pt_br/freertos/latest/userguide/freertos-account-and-permissions.html>.

22 _____. **Configurar um Raspbery Pi - AWS IoT Greengrass**. Disponível em: <https://docs.aws.amazon.com/pt_br/greengrass/latest/developerguide/setup-filter.rpi.html>.

23 _____._____. Disponível em: <https://docs.aws.amazon.com/pt_br/greengrass/latest/developerguide/gg-device-start.html>.

24 _____._____. Disponível em: <https://docs.aws.amazon.com/pt_br/greengrass/latest/developerguide/gg-config.html>.

25 _____. **Fluxo de dados do serviço Device Shadow**. Disponível em: <https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/device-shadow-data-flow.html>.

26 SERVICES, Amazon Web. **Guias de conceitos básicos específicos da placa**. Disponível em: <https://docs.aws.amazon.com/pt_br/freertos/latest/userguide/getting-started-guides.html>.

27 _____. **Interagir com device shadows**. Disponível em: <https://docs.aws.amazon.com/pt_br/greengrass/latest/developerguide/modulez.html>.

28 _____. **O que é o Amazon FreeRTOS**. [S.l.: s.n.]. Disponível em: <https://docs.aws.amazon.com/pt_br/freertos/latest/userguide/what-is-amazon-freertos.html>. Acesso em: 25 jun. 2019.

29 _____. **O que é o AWS IoT GreenGrass**. [S.l.: s.n.]. Disponível em: <https://docs.aws.amazon.com/pt_br/greengrass/latest/developerguide/what-is-gg.html>. Acesso em: 25 jun. 2019.

30 _____. **Registro da placa MCU com AWS IoT**. Disponível em: <https://docs.aws.amazon.com/pt_br/freertos/latest/userguide/get-started-freertos-thing.html>.

31 _____. **Serviço Device Shadow para AWS IoT**. Disponível em: <https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/iot-device-shadows.html>.

32 STANKOVIC, John A.; RAJKUMAR, R. Real-Time Operating Systems. **Real-Time Syst.**, Kluwer Academic Publishers, Norwell, MA, USA, v. 28, n. 2-3, p. 237–253, nov. 2004. ISSN 0922-6443. DOI: 10.1023/B:TIME.0000045319.20260.73. Disponível em: <https://doi.org/10.1023/B:TIME.0000045319.20260.73>.

33 SYSTEMS, Espressif. **ESP32 Sereies Datasheet**. [S.l.: s.n.], 2019. Disponível em: <https://www.espressif.com/en/support/download/documents>. Acesso em: 25 jun. 2019.