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

JARDEL OSORIO DUARTE

ANÁLISE DE DESEMPENHO DO RASPBERRY PI SENDO UTILIZADO COMO BROKER DO PROTOCOLO MQTT

CHAPECÓ
2021

JARDEL OSORIO DUARTE

ANÁLISE DE DESEMPENHO DO RASPBERRY PI SENDO UTILIZADO COMO BROKER DO PROTOCOLO MQTT

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: Prof. Dr. Marco Aurelio Spohn

CHAPECÓ
2021

Duarte, Jardel Osorio

Análise de desempenho do Raspberry Pi sendo utilizado como broker do protocolo MQTT / Jardel Osorio Duarte. – 2021.
49 f.: il.

Orientador: Prof. Dr. Marco Aurelio Spohn.
Trabalho de conclusão de curso (graduação) – Universidade Federal da Fronteira Sul, curso de Ciência da Computação, Chapecó, SC, 2021.

1. rede IoT. 2. Inspecionar. 3. *Protocolo MQTT*. 4. Rasberry Pi. I. Spohn, Prof. Dr. Marco Aurelio, orientador. II. Universidade Federal da Fronteira Sul. III. Título.

E-mail: jardelduarte594@gmail.com

JARDEL OSORIO DUARTE

ANÁLISE DE DESEMPENHO DO RASPBERRY PI SENDO UTILIZADO COMO BROKER DO PROTOCOLO MQTT

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: Prof. Dr. Marco Aurelio Spohn

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em: 4/10/2021.

BANCA AVALIADORA

Prof. Dr. Marco Aurelio Spohn – UFFS

Prof. Me. Adriano Sanick Padilha – UFFS

Profa. Dr. Braulio Adriano de Mello – UFFS

RESUMO

Atualmente a utilização da rede IoT é uma realidade dentro da sociedade, isto se dá em consequência da ubiquidade gerada através de projetos de cidades inteligentes e indústria 4.0, estas coisas possibilitaram abranger em paralelo novos espaços para uma vasta gama de operacionalidades no serviço global de internet, exemplos como protocolos de rede, servidores em nuvem, exploração e sanitização de dados, entre outros serviços compartilhados. Existe um imenso horizonte de possibilidades que surgem através de estudos em redes IoT, consequentemente por ser composta de objetos comuns relacionados à vida humana onde é possível idealizar diversas atividades vinculadas ao bem estar social em que uma aplicação ajudaria a desempenhar. Em suma, para que haja uma validação nestas pesquisas é necessário inspecionar todos os componentes com propósito de assegurar um serviço confiável nas respectivas implementações, e em virtude disso, a presente pesquisa contribui com a extração de dados analisados a partir de um broker (servidor) do *protocolo MQTT* implementado no dispositivo Raspberry Pi, com a intenção de experenciar com eventuais disparos de publicadores e assinantes para obter a relação dos índices de consumo de memória, capacidade de processamento e detecção de perda de pacotes na rede.

Palavras-chave: rede IoT. Inspecionar. *Protocolo MQTT*. Rasberry Pi.

ABSTRACT

Currently, the use of the IoT network is a reality within society, consequently because of the ubiquity generated through smart city projects and industry 4.0, these things have made it possible to revolutionize new spaces for a wide range of operations in the global internet service, examples such as network protocols, cloud servers, data exploration and data sanitization, among other shared services. An immense horizon of possibilities arises through studies in IoT networks, for being composed of common objects related to human life it is possible to idealize various activities linked to social welfare in which an application would help to perform. For this reason, for validation it is necessary to inspect all components in order to guarantee a reliable service in certain implementations. And as a result, this research contributes to the extraction of analyzed data from a broker (server) of the *MQTT protocol* implemented in the Raspberry Pi device, with the intention of exploring, with eventual triggers from publishers and subscribers the behavior of the device in terms of its memory utilization, processing capacity and detection of packet loss in the network.

Keywords: IoT networks. analyzed data. *MQTT protocol*. Raspberry Pi.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus pela possibilidade de vida diante de um momento tão triste para a história do mundo, em principal do Brasil que contabilizou um dos maiores índices de mortes por Covid-19 nos últimos tempos. Agradeço aos meus familiares, em principal minha mãe Ana Cristina Osório Duarte pelo amparo e apoio diante todo o período universitário, e que embora não estando presente em todos os momentos nunca me deixou faltar nada, aos professores que partilharam todos os ensinamentos, principalmente ao meu orientador Prof. Dr. Marco Sphon que possibilitou o desenvolvimento deste trabalho, além de outros diversos conhecimentos.

Agradeço aos colegas que além de trilharem lado à lado almejando o título de bacharel em ciência da computação foram fundamentais dando suporte psicológico e incentivando em muitos momentos desta caminhada, aos estudantes do pró-Haiti pelo compartilhamento cultural e amizade, também agradeço aos demais estudantes da Universidade Federal da Fronteira Sul que de forma ou outra contribuíram seja nas disciplinas de tronco comum ou em eventos interdisciplinares com a formação de um caráter humanizado e uma personalidade mais íntegra. Por fim, agradeço aos meus amigos que em muitos momentos foram fundamentais para que eu suportasse os problemas que surgiram mediante a todos estes anos de estudos.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

IoT Internet of Things

MQTT Message Queue Telemetry Transport

MIT Massachusetts Institute of Technology

M2M Machine to Machine

WIFI Wireless Fidelity

ITU Telecommunication Union

TCP Transmission Control Protocol

UDP User Datagram Protocol

IP Internet Protocol

CoAP Constrained Application Protocol

RFC Request For Comments

XMPP Extensible Messaging and Presence Protocol

XML Extensible Markup Language

IETF Internet Engineering Task Force

RFID Radio-Frequency Identification

FCC Federal Communications Commission

IPv6 Internet Protocol version 6

IIoT Industrial Internet of Things

SSL Secure Sockets Layer

TLS Transport Layer Security

API Application Programming Interface

IoT-A Internet of Things Architecture

VM Virtual Machine

AWS Amazon Web Service

GCP Google Cloud Platform

IBM International Business Machines Corporation

IaaS Infrastructure as a Service

PaaS Platform as a Service

DCaaS Data Centers as a Service

OASIS Organization for the Advancement of Structured Information Standards

SDK Software Development Kit

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

A Internet of Things (IoT) surgiu em 1999 através do Massachusetts Institute of Technology (MIT) e desde então o conceito é propriamente atribuído a comunicação Machine to Machine (M2M), ou seja, máquinas se comunicando entre si, em geral sensores, conexão de eletrônicos, controladores e diversos outros atuadores através de redes sem fio Wireless Fidelity (WIFI) e/ou cabeada. Estes dispositivos em sua grande parte se comunicam em cima de uma camada de rede utilizando de protocolos e servidores na nuvem para armazenar e disseminar as informações publicadas. Somando a simplicidade da aplicação com a quantia de conexões suportadas e o baixo consumo de energia dos sensores, a rede IoT cresceu de uma maneira exponencial nos desenvolvimentos de projetos de mobilidade urbana, residencial e industrial.

A Telecommunication Union (ITU) traz que a internet das coisas é, *"Uma infraestrutura global para a sociedade da informação, permitindo serviços avançados através da interconexão (física e virtual) de coisas baseadas em tecnologias interoperáveis de informação e comunicação, existentes e em evolução ".* (10) Em uma análise estimativa a IoT tem probabilidade de crescer com ganhos de 25 a 50 bilhões de dispositivos conectados até 2025. Manyika traz que as principais áreas de potencial investimento são saúde, agricultura e fabricantes que utilizam sensores para otimizar a manutenção de equipamentos e proteger a segurança dos trabalhadores. Em sua análise ascendente é dimensionado um impacto econômico com potencial total de U$3,9 trilhões a U$11,1 trilhões por ano até 2025, o que seria aproximadamente 11% da economia mundial. (12).

Para que haja a possibilidade de uma comunicação neste modelo é necessário que exista uma arquitetura de rede que permita o envio e o recebimento de todas essas informações. Os pacotes de dados necessitam de protocolos para serem entregues em seus destinos e essa comunicação é feita através de protocolos como Transmission Control Protocol (TCP), User Datagram Protocol (UDP), entre outros. Tanto o TCP quanto o UDP possuem a garantia de entrega e enfileiramento de mensagens e funcionam em cima do Internet Protocol (IP). O IP é responsável pelo endereçamento das máquinas garantindo que estas estejam identificadas na rede. Outros protocolos bastante comuns em rede IoT são; o *protocolo Message Queue Telemetry Transport (MQTT)*, aplicado em cima do TCP/IP que utiliza de agentes mensageiros nomeados como publisher e subscriber que se comunicam mediante a um broker atuante como um servidor; o Constrained Application Protocol (CoAP) que é um protocolo de padrão Request For Comments (RFC) e é utilizado em aplicação de sensores tendo em vista a simplicidade, porém este possui limitação de conexão e processamento; o Extensible Messaging and Presence Protocol (XMPP), um protocolo aberto baseado em Extensible Markup Language (XML) com padrão Internet Engineering Task Force (IETF) projetado para uso de aplicações de mensagens instantâneas em tempo real (6); e outros vários.

Completando o cenário da comunicação M2M também é fundamental que haja um gateway, pois é ele o intermediário que faz a distribuição dos pacotes de mensagens lançados por cada sensor através de seus protocolos, também podendo ser nomeado como um tradutor de protocolos nesta mediação. É comum encontrar nessa camada os dispositivos de baixo custo com sistemas embarcados, exemplos desses hardwares são o Arduino, Beaglebone, Raspberry, etc... O gateway no IoT está agregado ao conceito de Fog Computing (Computação de borda ou computação em névoa) pois se encontra exatamente entre os sensores e a Cloud (nuvem) e permitem que mais coisas sejam introduzidas em operações de conexão em escala industrial, isto por suportarem diversos protocolos ou serem implementados como multiprotocolos devido a aplicação em cenários difíceis da indústria 4.0 e cidades inteligentes.

Entretanto, Silva comenta em sua monografia que encontram-se desafios para o futura da IoT nos próximos anos, exemplos como a regulamentação, a padronização e a segurança, isto por motivo de ter muitos protocolos implementados para rede IoT, onde cada protocolo tende a resolver somente um problema, ou seja, se mantém restrito quanto à utilização geral. Ele também fomenta que em um curto espaço de tempo a estimativa é que o IoT será incluído em quase todos os dispositivos móveis, sendo estes medidores de estacionamento, pneus, estradas, monitores cardiovasculares, termostatos, sensores de presença, prateleiras de supermercados e uma imensa quantia de dispositivos conectados com a internet das coisas, gerando uma propensa exploração das vulnerabilidade, problemas de privacidade com a disseminação dos dados sensíveis e segurança com possíveis cenários de ataques. Para isso precisamos de regulamentos adequados e organizados tanto para evitar tais riscos de segurança como para lidar com os riscos de todos os dispositivos IoT conectados.(20)

Enfim, a partir da análise do crescimento da internet das coisas, surge a proposta visando explorar alguns destes exemplos citados, em principal o *protocolo MQTT* e o Raspberry Pi, com intuito de analisar custos de processamento, o índice de utilização de memória e o tráfego da rede. Onde o cenário é determinado por um dispositivo Raspberry Pi sendo responsável pelo serviço do corretor/broker e um computador fazendo disparos de publicações e assinaturas, para então explorar o gargalo do dispositivo e resultar os dados obtidos nestes eventos.

1.2 PROBLEMÁTICA

Partindo das demandas provindas da área de telecomunicação e redes e da visível curva de crescimento da internet das coisas, fica claro através da bibliografia a necessidade de examinar cada dispositivo a ser implementado como broker ou servidor, razões estas que possam garantir a integridade na execução dos possíveis cenários, principalmente sabendo que estas implementações estão diretamente atribuídas a vida humana, saúde em geral e em cidades inteligentes. Para validar estas pesquisas também é importante que o tratamento dos dados sensíveis possuam todos os métodos que contenham os pilares da segurança de informação (confiabilidade, integridade e disponibilidade), prevenindo acidentes que possam levar à risco o bem estar social

(sendo os riscos: carros autônomos, casas com sensores inteligentes, monitores cardiovasculares e outros diversos) ou a validade de um estrutura que hoje já está consolidada. Com base nestas apostas, surgem os desafios nas diferentes opções de hardwares de distintas arquiteturas e corporações, os variados protocolos de rede e outros aspectos, sendo então necessário a realização de benchmarking para uma análise aprofundada destes resultados, identificando o melhor a ser adotado em cada cenário simulado. Nestes testes de performance são incluídos alguns problemas para garantir a confiabilidade, por exemplo o índice de perda de pacotes, capacidade de transferência de dados, capacidade de processamento e uso de memória. Permitindo então uma estatística aproximada da capacidade daquele hardware. Neste trabalho, foi escolhido o Raspberry Pi como dispositivo a ser analisado, onde a proposta é a implementação de um broker Mosquitto no dispositivo com a finalidade de assegurar a capacidade do Raspberry Pi através de um benchmarking, fazendo uso de softwares em tempo real para acompanhar o consumo de processamento durante a escalabilidade de requisições, o consumo memória e a quantidade de pacotes perdidos na rede.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

O presente trabalho objetiva investigar a capacidade do Raspberry Pi utilizando o broker do *protocolo MQTT*, onde o foco baseia-se na análise de possíveis cenários, gerando disparos de publicadores e assinantes conectados por apenas um servidor/broker.

2.2 OBJETIVOS ESPECÍFICOS

- Implementar o broker Mosquitto no Raspberry Pi;
- Acompanhar a capacidade de processamento, o consumo memória e perda de pacotes na rede;
- Fazer testes com rede cabeada e rede sem fio;
- Testar diferentes cenários(um-para-muitos, muitos-para-muitos e muitos-para-um);
- Analisar os testes de cada cenário;
- Resultar os melhores cenários comparado aos demais;

3 JUSTIFICATIVA

Apresenta como relevância uma analise aprofundada do dispositivo Raspberry Pi em resposta ao broker Mosquitto, experenciando o modelo publisher-subscriber do *protocolo MQTT*, a fim de resultar o desempenho do hardware apontado, sobretudo considerando cenários homogêneos de casos de uso, buscando garantir a capacidade do Raspberry Pi como um gateway estável em cenários de numerosas conexões em rede IoT. Estrutura que atende atualmente milhares de requisições simultâneas (leituras, publicações de mensagens) assíncrona entre as partes. Por fim, espera-se com a pesquisa identificar resultados positivos e encorajar mais análises com viez ao tema.

4 REVISÃO BIBLIOGRÁFICA

4.1 IOT

Kevin, especialista britânico do MIT traz a definição para IoT sendo *"um novo mundo em que os objetos estarão conectados e passarão a realizar tarefas sem a interferência humana"*. (3) Pode-se considerar uma simples explicação para IoT sendo conexões entre pessoas e máquinas, ou também máquinas com máquinas M2M. Atualmente, sensores são incluídos em todos os lugares e estes sensores convertem dados físicos brutos em sinais digitais que são transmitidos ao seu centro de controle, permitindo monitorar mudanças climáticas, controle de tráfego, análise preditiva de saúde, etc. Esta arquitetura foi baseada pensando no contexto de operações e processos em cenários de tempo real.

Suresh incute que o IoT pode estar sendo implementado na automação residencial, conectando a caixa de distribuição de energia a um único smartphone (ou a controlador) em que poderia estar sendo operado remotamente, na industrial, através de um dispositivo embarcado servindo como gateway, em que em ambos os cenários não seria necessário um dispositivo local de armazenamento instalado para que o evento ocorra, podendo então um único sensor capturar sinais e processá-los encaminhando-os a serviços compartilhado na internet, porém esta arquitetura varia dependendo do contexto de sua aplicação.(21)

4.1.1 História

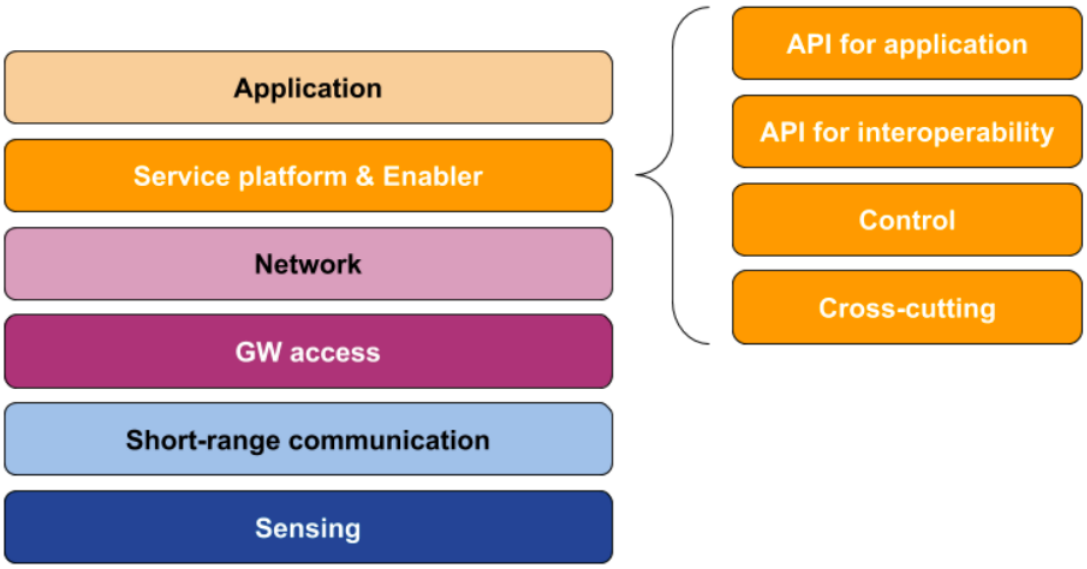
Desde o surgimento da internet em 1989 estamos conectando coisas na internet, e além dos objetos de interação humana direta (computadores pessoais), neste primeiro momento ocorreram testes de dispositivos eletrônicos conectados, como a cafeteira Trojan Room sendo citada por Suresh como sendo o primeiro dispositivo aplicado nesses cenários, logo em seguida, em 1990 John Romkey criou uma torradeira que tournou-se o primeiro dispositivo a ser ligada e desligado através da internet, e em 1994 surge o WearCam inventado por Steve Mann que usando um sistema de 64 processadores foi o primeiro computador vestível que tinha como objetivo o envio de imagens pencigráficas para uma estação base no telhado de seu edifício. Em 1997 o sensoriamento já virava uma realidade, Paul Saffo's trouxe a descrição sobre os sensores e seu futuro curso de ação para o ano, e em 1999 o termo IoT nascia através de Kevin Ashton, diretor executivo do AutoIDCentre, MIT, ainda em 1999 eles também viabilizaram o item global Radio-Frequency Identification (RFID) baseado em sistemas de radares de aviões da segunda guerra mundial. Após estes eventos ocorreu um grande salto na comercialização da Internet das Coisas,e em 2000 a LG anunciou o interesse em desenvolver uma geladeira inteligente que determinaria o momento de seu reabastecimento, em 2003 o RFID foi implementado em grande escala no exército dos EUA durante o programa Savi, no mesmo ano o Walmart varejo implementou RFID em todas as sua lojas distribuídas no mundo. (21)

O tema ganhou ainda mais destaque em 2005 com os publicadores The Guardian, Scientific American e Boston Globe citando em diversos artigos a relevância da IoT e seu curso futuro, dando visibilidade para posteriormente grupos de empresas lançarem o IPSO Alliance para promover o uso do Protocolo de Internet (IP) em redes de objetos inteligentes, viabilizando ainda mais o IoT. Enfim, em 2011 após a Federal Communications Commission (FCC) aprovar o uso do espectro de espaço em branco foi lançado o Internet Protocol version 6 (IPv6) que desencadeou um crescimento massivo de interesses neste campo. (21)

4.1.2 Arquitetura IoT

Embora não exista uma representação que esboce uma arquitetura homogênea da IoT, Di Martino et al. traz em seu artigo que recentemente algumas pesquisas propuseram uma definição de rede IoT delineando uma maneira interessante de representar a pilha tecnológica, ou seja a arquitetura da rede proposta por Borgia. (7)

Figura 1 – Arquitetura IoT de seis camadas



Fonte: Di Martino et al. (7)

A arquitetura proposta é representada na figura 1 e está composta em seis camadas, onde são definidas da seguinte forma:

- **Atuadores (Sensing):** Detecção através dos sensores;
- **Conectividade (short-range communication):** Comunicação de curto alcance ou conectividade do sensor através de bluetooth, rede sem fio, RFID, outros;
- **Acesso ao gateway (GW access):** Controlador de sistema embarcado;

- **Conexão de alta largura de banda (Network):** Acesso à rede internet, ocorre de forma cabeado ou de rede sem fio;

- **Tratamento dos dados (Service Platform e enabler):** A quinta camada é o serviço de tratamento dos dados, incluindo softwares e serviços dedicados ao controle oferecidos pelas plataformas;

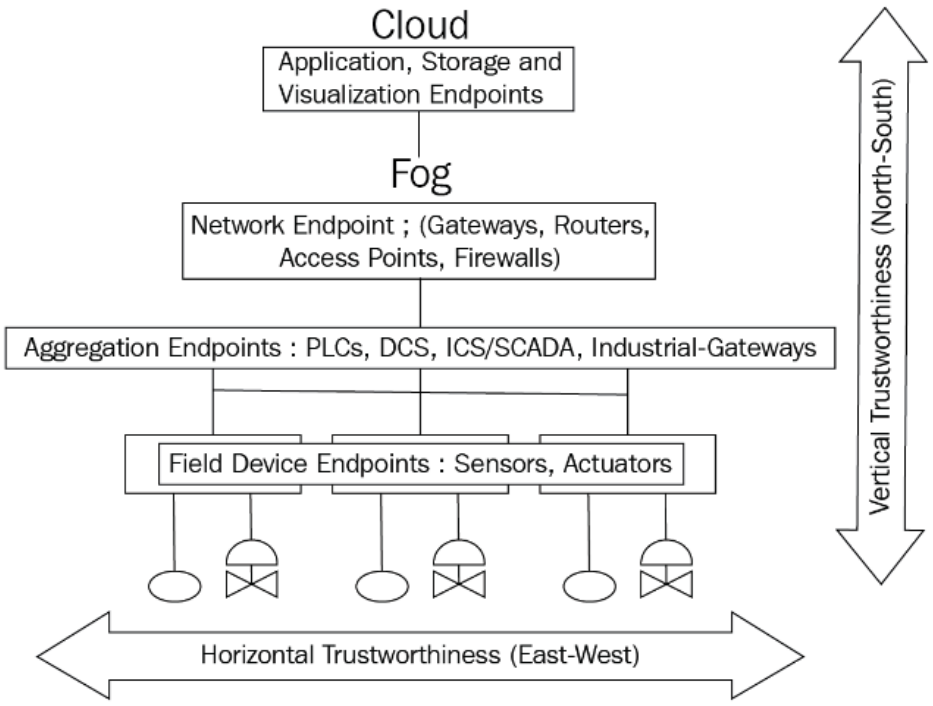
- **Camada de aplicação (Application):** Por fim a camada de aplicação, ou seja, serviços de domínio, para onde enviar os dados tratados ou mantê-los.

Em outra visão, Bhattacharjee traz em seu livro que é muito importante arquiteturas confiáveis para Industrial Internet of Things (IIoT), vide ser uma tecnologia-chave na indústria 4.0 que revolucionou a maneira como fábricas e organizações industriais se desenvolvem. Focado em uma arquitetura de endpoints(pontos finais e/ou de ponto a ponto) baseada em risco, o autor trouxe dois princípios importantes que envolvem motivação e análise de risco. (5)

- **Motivação:** As motivações mais comuns são a proteção a segurança, confiabilidade, resiliência e privacidade dos dispositivos, garantindo integridade e disponibilidade.

- **Análise de risco:** Riscos como explorações, usando dispositivos IoT (bots) como vetores de ataque, risco a garantia de vida, e outros.

Figura 2 – Arquitetura endpoints IIoT

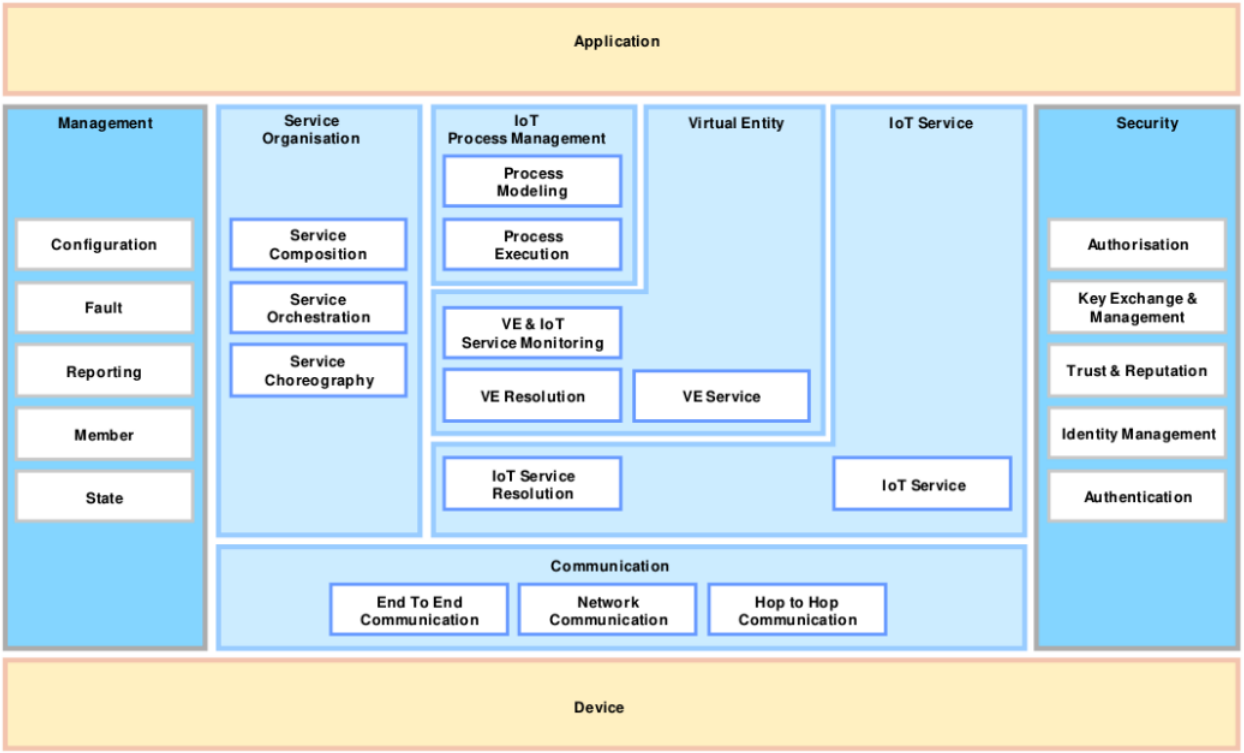


Fonte: Bhattacharjee (5)

Considerando que estas aplicações envolvem custos e podem consumir muitos recursos, a segurança de um sistema corporativo industrial não pode estar totalmente associada a firewalls e gateways tunelados por protocolos de segurança nas topologias convencionais. Bhattacharjee trouxe a estratégia de proteção de endpoints com boa relação no modelo de quatro camadas, fomentando a utilização de criptografias em todos os blocos da arquitetura para a criação delineada de um ambiente confiável de ponta a ponta, sendo então a criptografia do hardware, protocolos criptografados, armazenamentos criptografados e técnicas de isolamento os métodos mais adotados.

No IIoT, os desafios encontrados também são devido a dados atravessarem limites organizacionais, os acessos remotos de interfaces terminais de outros domínios organizacionais, e os endpoints dos dispositivos vem de uma variedade de fornecedores. No entanto, recursos de nível de Application Programming Interface (API) consistentes em todos os terminais, tanto na borda quanto na nuvem e a adesão a arquiteturas modulares e escaláveis podem simplificar o projeto geral de segurança. A figura 2 mostra vários pontos de verificação no ciclo de vida do modelo endpoints. (5)

Figura 3 – IoT-Architecture



Fonte: Bassi et al. (4).

Existem também arquiteturas que esboçam outros cenários na IoT que vão além dos padrões abstratos, um exemplo é a Internet of Things Architecture (IoT-A) visto na figura 3, o IoT-A é uma fundação da comissão europeia em que empresas podem capitalizar sobre os benefícios do desenvolvimento de plataformas orientadas para o consumidor, incluindo serviços,

hardware e software. No projeto European Lighthouse Integrated Project foi apresentado um modelo arquitetônico construído durante três anos (de setembro de 2010 até novembro de 2013), que propôs a criação de uma arquitetura como modelo de referência junto a definição de um conjunto de blocos de construção chave, vistos como fundamentais para promover uma futura Internet das Coisas. Partindo de um paradigma experimental o IoT-A combinou o raciocínio de cima para baixo sobre os princípios arquitetônicos e as diretrizes de design com simulação e prototipagem focados em consequências técnicas do projeto. O projeto previu resultados como interoperabilidade; delinear conceitos abstratos para o projeto e seus protocolos, interfaces e algoritmos; mecanismos eficientes na integração de serviços para internet do futuro; infraestrutura de resolução, buscas e descobertas escalonáveis de captação de recursos da IoT, entidades reais e associações; novos componentes da plataforma. (17)

4.1.3 Rede IoT utilizando *protocolo MQTT*

A adoção do *protocolo MQTT* na IoT cresceu muito nos últimos anos vide a fácil utilização e o suporte a mensagens através de um broker, onde a complexidade é tão baixa que a simples configuração e instalação de broker MQTT utilizando uma Virtual Machine (VM) já é suficiente para uma comunicação existir. Essa desenvoltura permite a execução de projetos residenciais serem desenvolvidos por um simples entusiasta, exemplo comumente visto é o monitoramento de luzes em partes de uma casa.

Em suma essa facilidade também é encontrada na literatura, durante as pesquisas para realização deste trabalho foram encontrados artigos, monografias, livros em diferentes áreas da ciência (saúde, engenharia, elétrica e outras), obordando o desenvolvimento de rede IoT em paralelo com o *protocolo MQTT*, um exemplo que chamou a atenção foi a monografia de Muenchen que teve como objetivo a aplicação de um sistema capaz de detectar fumaças e diferentes gases inflamáveis através de uma placa ESP-32 e um sensor de fumaça MQ2, o software implementado tinha funções de hibernação visando o baixo consumo de energia quando estes não entravam em cenários de ameaça. Em síntese este projeto idealizou o encaminhamento das publicações de indícios de incêndios de casas de show, bares, lojas de varejo e outros serviços de atendimento. Nesta pesquisa o MQTT foi o protocolo adotado para o envio das mensagens diretamente para nuvem, o que garantiria um melhor retorno do serviço de segurança pública sabendo que as mensagens seriam entregue em tempo real, bastando somente um computador conectado e um broker instalada para assinar os tópicos e receber as notificações de cada entidade. Inclusive Muenchen também comenta a necessidade de uma configuração de geolocalização nos dispositivos para melhor assertividade e a implementação de funções no gateway que garantissem o funcionamento imediato dos sinalizadores de saída acusando o risco eminente. (14).

Enfim, o *protocolo MQTT* é visto como um padrão de rede IoT e ao pensarmos nas inúmeras possibilidades de aplicações, surge um horizonte diante dos olhos visando uma melhor qualidade de vida.

4.1.4 Cloud Computing

Segundo a página PET de Sistemas de Informação da Universidade Federal de Santa Maria(UFSM) o termo Cloud Computing foi definido pelo professor de sistemas de informação Ramnath Chellappa em 1997 durante uma palestra universitária apresentando a definição a algo que está no ar, fazendo uma relação com sistemas que não estão hospedados em servidores físicos, mas que está na rede. (15) Acabando com parte dos desafios que o IoT ainda tinha em 2000, a Cloud Computing chega ao amplo público com a Salesforce e com a gigante Amazon, estes primeiros modelos de serviços consistiam em aluguéis de computadores virtuais(VMs), qual disponibilizavam seus próprios serviços e aplicativos da plataforma, logo em seguida vieram também as nuvens Google e Microsoft que ofertavam serviços adicionais com custos menores. Entretanto, somente em 2006 esse modelo de serviços se consolidaram em plataformas totalmente focadas a manipulação de computadores compartilhados, exemplos como a Amazon Web Service (AWS), ou em 2007 com a Netflix(serviço de streaming de dados em nuvem que possibilita aos assinantes assistir filmes e séries estando somente conectado à rede), em 2008 com a Google Cloud Platform (GCP) e em 2010 com o Watson International Business Machines Corporation (IBM) e Azure da Microsoft. (15)

Deve ser levado em consideração a grande manipulação de dados da IoT um dos pilares do modelo compartilhado em nuvem, razões que podem ser facilmente atribuídas ao fato de dispositivos de baixo custo (sensores, controladores, etc.) não possuírem armazenamento em grande escala, sendo fundamental que a nuvem sirva de armazenamento, hospedagem e replicação das mensagens publicadas por estes controladores, entretanto existem inúmeros serviços ofertados nessas plataformas onde é importante também salientar características de integridade, confiabilidade, flexibilidade e escalabilidade dos dados.

A comunicação dos processos a partir dos serviços ofertados em centros de processamento de dados compartilhados(data centers), ocorrem de forma simples. Entre os serviços destacados para rede IoT estão, o armazenamento dos dados na própria plataforma e a virtualização dos dados nas VMs, ou seja, a máquina operando como broker na nuvem encaminhando as mensagens recebidas por determinados protocolos de rede, evitando a necessidade de um servidor físico.

Segundo o setor de dados da Associação Brasileira de Empresas de Software (ABES), os gastos de Infrastructure as a Service (IaaS) somados com de Platform as a Service (PaaS) em nuvem pública no Brasil devem atingir em 2021 cerca de US$ 3,0 bilhões, o que representa um crescimento da adoção em 46,5% em relação à 2020. O modelo de nuvem privada também cresceu em um bom ritmo, totalizando US $614 milhões neste ano, crescimento vide ao Data Centers as a Service (DCaaS) que avançou 15,5% comparado à 2020. (1). Enfim, para complementar o cenário de cloud uma breve definição das nuvens existentes atualmente.

- **Nuvem Pública:** A nuvem pública pode ser definida como uma grande oferta de serviços de computação concedida por determinadas empresas (terceirizadas) à internet, em alguns

casos podem ser serviços gratuitos ou vendidos sob demanda, permitindo que os clientes paguem apenas o consumo relacionado ao serviço contratado, exemplos são os ciclos de processamento, armazenamento e largura de banda.

- **Nuvem Privada:** A nuvem privada consiste na oferta de recursos de computação direcionados a atender exclusivamente uma única empresa, a arquitetura do data center e a manutenção dos servidores é totalmente exclusiva tanto para o planejamento quanto para execução. Esse modelo costuma ser adotado por instituições financeiras, órgãos governamentais, e empresas de grande porte que possuem críticas aos negócios, razões de adoção e melhor controle sobre o ambiente e segurança.

- **Nuvem Híbrida:** O modelo de serviço híbrido tem finalidade de somar a nuvem pública com a nuvem privada, podendo oferecer tanto um gerenciamento exclusivo mas adotando diversas operacionalidades de serviços públicos, além de trazer o conceito de várias nuvens, opção atualmente ofertada pela Google Cloud Platform. Em um artigo da GCP (2021) os objetivos gerais abordados do modelo híbrido de várias nuvens é atender aos requisitos de execução de cargas de trabalho para o ambiente computacional; adoção estratégia de qual padrão será aplicado às diferentes cargas de trabalho; qual tecnologia e topologia de rede a ser usada. Adotando um processo cíclico buscando a influência onde cada requisito tende sobre o outro, como um sistema de treinamento garantindo melhor desenvoltura. (8)

4.2 *PROTOCOLO MQTT*

Segundo a página oficial do *protocolo MQTT*, este protocolo foi criado pelo Dr. Andy Stanford-Clark da IBM e Arlen Nipper da Arcom(agora Eurotech), em 1999. (13) O atual CEO da Huawei Michael Yuan trouxe em um artigo publicado na IBM Developer que o *protocolo MQTT* foi inicialmente implementado com o objetivo de atuar em cenários heterogêneos da IoT, e neste primeiro momento era vinculado a aplicações em pipeline de petróleos a satélites, entretanto logo em seguida tornou-se o padrão para comunicação na rede IoT. (24)

Yuan comenta que o *protocolo MQTT* é um protocolo de mensagem com suporte de comunicação assíncrona entre as partes (emissor e receptor) e portanto é escalável em ambientes de redes não confiáveis, ele não possui relação com modelos de enfileiramento de mensagens, mas adota um modelo de publicação (publisher) e assinatura (subscriber) utilizando de um broker em cima do TCP/IP que possui enfileiramento. No final de 2014 o MQTT tornou-se oficialmente um padrão aberto da Organization for the Advancement of Structured Information Standards (OASIS) e assim possibilitou suporte a linguagens de programação populares atualmente, o que garantiu implementações de softwares livres. (24)

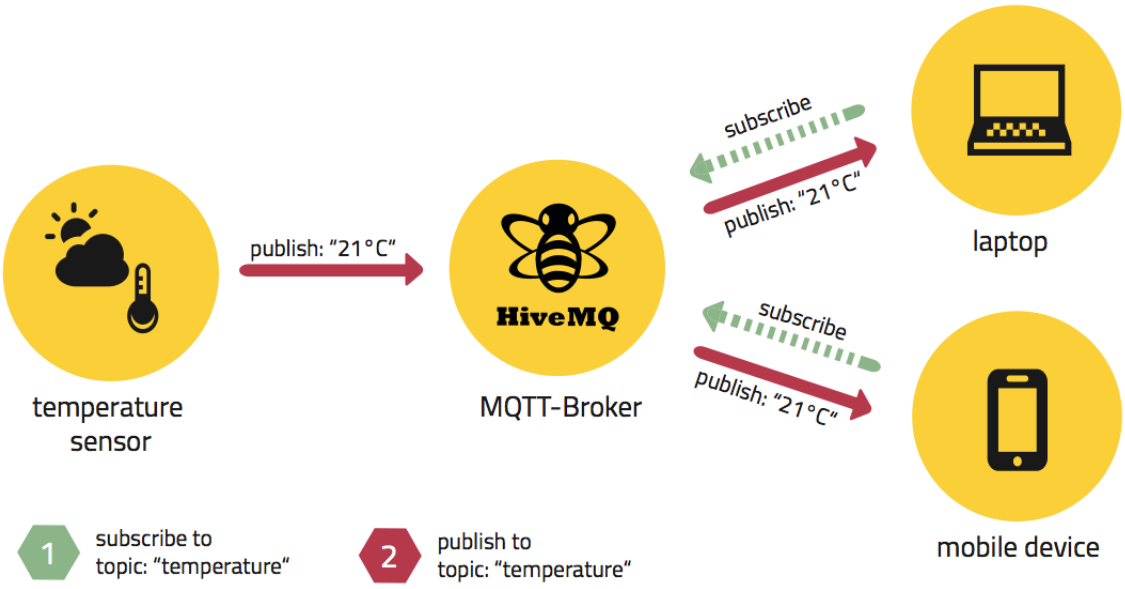
A página oficial do *protocolo MQTT* também traz um conceito relacionado a garantia de trocas de mensagens seguras à partir deste protocolo, entretanto na aplicação com estes padrões

o usuário deve informar o nome e uma senha com um pacote MQTT onde a criptografia na rede pode ser tratada com Secure Sockets Layer (SSL) ou atual Transport Layer Security (TLS) (embora o SSL não seja o mais leve dos protocolos e adicionaria sobrecarga significativa na rede), a documentação reconhece também modelos de seguranças adicionais de aplicativos embora não embutidos no protocolo a fim de mantê-lo simples e leve. (13)

4.2.1 Broker

O broker opera como um servidor, e como o MQTT funciona em um modelo de publicação e assinatura, é fundamental que todas as execuções sejam mediadas por ele. O broker então faz o roteamento das mensagens dos clientes publicadores e em seguida envia para os clientes assinantes. Segundo Yuan um cliente tanto pode ser um sensor de IoT em campo ou um aplicativo em um data center que processa dados de IoT. (24)

Figura 4 – Arquitetura Publish/Subscribe MQTT



Fonte: Götz (9)

O *protocolo MQTT* define dois tipos de entidades na rede: um mensageiro broker e inúmeros clientes (publishers e subscribers), a figura 4 mostra o exato instante em que um publicador do tópico "temperature"envia mensagens a dois clientes assinantes através de um broker.

- O cliente conecta-se ao broker. Ele pode assinar qualquer "tópico" de mensagem no broker. Essa conexão pode ser uma conexão TCP/IP simples ou uma conexão TLS criptografia para mensagens sensíveis.

- O cliente publica as mensagens em um tópico, enviando a mensagem e o tópico ao broker.
- Em seguida, o broker encaminha a mensagem a todos os clientes que assinam esse tópico.

4.2.2 Publisher & Subscriber

Para que haja a comunicação entre as pontas, tanto o publicador/publisher quanto o assinante/subscritor devem estar conectados a um broker. Em geral os clientes não têm um endereço específico padrão e não existem conexões direta entre eles. Existem diversos cenários que podem ser classificados, exemplos:

- Any to many: Um publicador para muitos assinantes ou muitos publicadores para um assinante;
- Any to any: Um publicador para um assinante;
- Many to many: Muitos publicadores para muitos assinantes;

4.2.3 Características de Mensagem

Atualmente existem diversos brokers de diferentes implementações do *protocolo MQTT* e cada um tem um modo de operar diferente, o broker Mosquitto é um exemplo, ele cria por definição um tópico somente de leitura com nome *$SYS*(tópico próprio do sistema), e então faz publicações de todos os logs de informação do serviço neste tópico, exemplos: timestamp; uptime; atividade dos clientes(se estão ativados ou desativados); todas as mensagens recebidas; todas as mensagens na memória.

O modelo de comunicação do *protocolo MQTT* é baseado em solicitação e resposta em que o pedido de autenticação ocorre através da solicitação do cliente (request) e uma resposta do servidor (response). O cabeçalho fixo das mensagens é composto por dois bytes como visto na figura a seguir.

Figura 5 – Cabeçalho de mensagem MQTT

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

Fonte: IBM; Eurotech (11)

A figura 5 representa a estrutura do cabeçalho em que o byte 1 é fragmentado em quatro partes, sendo 4 bits(de 7 à 4) reservados para o tipo de mensagem (Message Type); DUP flag no bit 3; o QoS level nos bits(2 e 1); e por fim o RATAIN flag no bit 0;

O byte 2 representa o número de bytes restantes na mensagem atual, incluindo dados no cabeçalho variável e a carga útil (payload). O esquema de codificação de comprimento variável usa um único byte para mensagens de até 127 bytes.

- **DUP flag (Duplicate Delivery):** É o sinalizador que deve ser definido quando o cliente ou servidor tentar entregar novamente uma mensagem PUBLISH, PUBREL, SUBSCRIBE ou UNSUBSCRIBE, só ocorre quando o QoS level é maior que zero (0).
- **QoS level(Quality of Service):** Definição do tipo de garantia de entrega que o cliente pretende para uma mensagem que pode ser definida em três níveis; O nível 0 consiste em não receber garantias (fire and forget); o nível 1 pelo uma vez a entrega é confirmada(Acknowledged delivery); e o nível 2 exatamente uma vez a entrega é garantida(Assured delivery).
- **RETAIN flag:** Garante a persistência, isso significa que quando um publicador envia uma mensagem ao broker com a flag retain ativada, esta mensagem será retida no broker e toda vez que um assinante se conectar ao tópico que contém a mensagem, ela será lançada.

Figura 6 – Conexão broker MQTT

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

Fonte: IBM; Eurotech (11)

A figura 6 traz uma definição do exato instante em que uma conexão acontece no *protocolo MQTT*.

- *CONNECT* - Quando uma conexão de soquete TCP/IP solicita conexão de um cliente para um broker, uma sessão de nível de protocolo é criada utilizando o fluxo Connect.
- *CONNACK* - O servidor envia uma mensagem em resposta a uma solicitação Connect de um cliente, podendo assumir valores como: (Conexão confirmada; Conexão recusada: versão de protocolo inaceitável; Conexão recusada: identificador rejeitado; Conexão recusada: servidor indisponível; Conexão recusada: nome de usuário ou senha incorretos; Conexão recusada: não autorizada; Reservado para uso futuro).
- *PUBLISH* - Cada mensagem Publish está diretamente associada a um único tópico(assunto ou canal) e quando enviada a um servidor é distribuída aos assinantes interessados neste mesmo tópico como uma mensagem publish.
- *PUBACK* - Uma mensagem Puback é a resposta de um servidor broker para uma Publish com QoS de nível 1.
- *PUBREC* - Uma mensagem Pubrec é a resposta de um servidor broker para uma Publish com QoS de nível 2.
- *PUBREL* - A pubrel é a resposta de um publicador a uma Pubrec do servidor ou a resposta do servidor a uma pubrec de um assinante, é a terceira mensagem no fluxo do protocolo QoS 2.
- *PUBCOMP* - Esta mensagem é a resposta do servidor a uma Pubrel de um publicador ou a resposta de um assinante de uma mensagem Pubrel do servidor, sendo a última mensagem no fluxo do protocolo em QoS 2.
- *SUBSCRIBE* - A mensagem Subscribe permite que um cliente registre interesses em um ou mais tópicos com o servidor broker e por consequência também é possível especificar o nível QoS em qual o assinante deseja receber as mensagens publicadas no tópico.
- *SUBACK* - Representa uma mensagem enviada pelo servidor ao cliente para confirmar o recebimento de uma mensagem Subscribe; a Suback contém uma lista de níveis de QoS concedidos.
- *UNSUBSCRIBE* - Representa uma mensagem enviada do cliente ao servidor para cancelar a assinatura de tópicos nomeados.
- *UNSUBACK* - Representa a resposta enviada pelo servidor ao cliente confirmando o recebimento de uma mensagem de cancelamento Unsubscribe.

- *PINGREQ* - A mensagem PINGREQ representa um "você está vivo?"enviada de um cliente conectado ao servidor, também conhecida como PING, “keep a alive”, “heartbeat”, entre outros.
- *PINGRESP* - Representa a resposta enviada por um servidor a uma mensagem PINGREQ e significa "sim, estou vivo".
- *DISCONNECT* - Representa uma mensagem enviada do cliente para o servidor para indicar que ele está prestes a fechar sua conexão TCP/IP de forma limpa e segura.(11).

4.3 TRABALHOS RELACIONADOS

4.3.1 Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo

Neste trabalho foram feitos estudos direcionados à analisar os brokers MQTT em suas variadas linguagens de implementações em sistemas embarcados e foi desenvolvido pelos integrantes do grupo de Redes de Computadores, Engenharia de Software e Sistemas(GREat) da Universidade Federal do Ceará (UFC), Andrei Torres, Atslands Rocha e José Neuman de Souza. O artigo foi publicado no 34° congresso da Sociedade Brasileira de Computação em 2016 e apresentado no XV Workshop em Desempenho de Sistemas Computacionais e de Comunicação.

Durante este projeto foi utilizado o Raspberry Pi 2 como gateway, onde buscou comparar os brokers de diferentes linguagens de programação (Java com o broker Apollo, Javascript com os brokers Mosca e Ponte, C com o broker Mosquitto e Erlang com o broker eMQTT).

O design experimental aplicado foi definido da seguinte ordem:

- Iniciar a captura dos dados;
- Aguardar 60 segundos;
- Iniciar 200 conexões de subscribers a cada 30 segundos;
- Ao atingir a carga máxima de 10.000 subscribers mantê-la durante 180 segundos;
- Encerrar 25 conexões a cada 1 segundo;
- Aguardar 5 minutos após encerrar a última conexão;
- Encerrar captura de dados.

Os resultados obtidos foram os seguintes, durante a comparação do processamento o eMQTT implementado em Erlang teve a maior carga de processamento provavelmente devido ao foco de aplicação em clusterização e escalabilidade, já o Mosca e Ponte implementados em Javascript nodejs (importante salientar que o Ponte implementa o MQTT à partir do Mosca sendo normal uma similaridade nas análises) obtiveram desempenhos equivalentes com o Mosquitto

em C, ambos consumindo menos de 25% do processamento total do dispositivo. Com referência ao consumo de memória, destaca-se novamente o Mosquitto consumindo somente 9,5 MB comparado ao consumo de memória de 275 MB do Mosca, Ponte e do terceiro colocado Earling. No que corresponde a rede foi analisado o índice de mensagens entregues (vazão de mensagens), o eMQTT alcançou o primeiro lugar seguidos do Ponte e Mosca, já o Mosquitto atingiu um índice de 68% de entregas aparentemente por ter atingido um pico máximo após 15 minutos (em torno de 4000 pacotes), o Apollo não obteve estabilidade em nenhuma fase dos testes e por isto não foi citado durante esta relação, acredita-se que por razões de memória.(22)

4.3.2 Análise de desempenho do BeagleBone Black utilizando o *protocolo MQTT*

Este trabalho teve como objetivo analisar o desempenho do BeagleBone Black operando como um gateway para IoT fazendo a utilização do *protocolo MQTT* e foi desenvolvido por Gabriel Augusto Veiga Rodrigues tendo como requisito à obtenção parcial do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Para execução dos testes foram utilizados o dispositivo BeagleBone Black como gateway e implementado no dispositivo o broker Mosquitto desenvolvido em C, foram realizados três fases de testes contendo um total de 6000 publishers/publicadores e analisado posteriormente o consumo de processamento, memória e perdas de pacotes na rede.

Cenário experimental do teste 1 (alcançar o nível máximo de carga no menor tempo):

- Iniciar coleta de desempenho;
- Aguardar 30 segundos;
- Iniciar com 0 conexões e aumentar 1200 conexões a cada 15 segundos;
- Ao atingir a carga máxima de 6 mil conexões, mantê-las durante 60 segundos;
- Encerrar 60 conexões a cada 1 segundo;
- Encerrar teste.

A duração do teste foi de 4 minutos e 33 segundos.

Cenário experimental do teste 2 (uso normal do gateway mantendo um balanço no total e tempo de conexões):

- Iniciar coleta de desempenho;
- Aguardar 30 segundos;
- Iniciar com 0 conexões e aumentar 300 conexões a cada 30 segundos;

- Ao atingir a carga máxima de 6.000 conexões mantê-la durante 90 segundos;
- Encerrar 30 conexões a cada 1 segundo;
- Encerrar teste.

A duração do teste foi de 16 minutos e 28 segundos.

Cenário experimental do teste 3 (teste de larga escala de tempo, tanto para alcançar a carga máxima como para mantê-la ativa):

- Iniciar coleta de desempenho;
- Aguardar 30 segundos;
- Iniciar com 0 conexões e aumentar 150 conexões a cada 30 segundos;
- Ao atingir a carga máxima de 6.000 conexões mantê-la durante 180 segundos;
- Encerrar 30 conexões a cada 1 segundo;
- Encerrar teste.

A duração do teste foi de 32 minutos e 58 segundos.

Os resultados alcançados foram, para os três testes o BeagleBone obteve um carga alta de processamento chegando a utilizar todo o processamento, onde nos primeiros 30 segundos houve uma estabilidade variando de 5% à 20% do uso, porém a partir da chegada do primeiro pacote o dispositivo utilizou 100% do processamento. Já no consumo de memória a distribuição foi da seguinte forma, no primeiro teste o pico máximo foi de 87 MB, no segundo teste o uso de memória teve pico de 97 MB alcançados no momento 151 à 170 (segundos de evento) e no último teste alcançou o pico máximo de 100 MB onde o consumo aumentava em média 14 MB em cada 30 segundos de evento. Por fim, nos dados relacionados a perda de pacotes, não houveram perdas a serem contabilizadas em nenhum dos eventos executados.(18)

5 METODOLOGIA

Durante a análise dos trabalhos relacionados fica notório que embora as pesquisas tenham ótimos resultados, ainda existem cenários a serem abordados e soma-se a isto a capacidade atual dos dispositivos. De outra forma, algumas lacunas em razão do alto processamento do Beaglebone Black e do comportamento errático do broker Mosquitto em relação a vazão de dados quando implementado no Raspberry Pi 2, motivando então uma análise ampliada dos cenários de testes que estão sendo detalhados neste capítulo.

5.1 FAMILIARIZAÇÃO COM O BROKER MOSQUITTO

Para um melhor entendimento, o Mosquitto é um módulo do projeto Pajo (desenvolvido em C), e a escolha foi composta em razão deste broker ser um broker de código aberto e sólido do *protocolo MQTT*, além de fornecer Software Development Kit (SDK)s e bibliotecas do MQTT em várias linguagens de programação, alguns testes já foram feitos com esta ferramenta para uma primeira avaliação de usabilidade, que incluiu a utilização do projeto Pajo, máquinas virtuais em nuvem e aplicações no sistema operacional Andriod para testes. Abaixo uma breve demonstração de uso do Mosquitto em um terminal linux trazida por Yuan. (24)

- Download e instalação o módulo Mosquitto no website mosquitto.org
- O comando mosquitto executa o broker do MQTT no computador local e é possível utilizar a flag *-d* para executar em segundo plano, então no terminal linux digite ***$ mosquitto -d***
- Em outra janela do terminal, é possível usar o comando mosquitto_sub para conectar-se ao broker local e assinar um tópico, e para isso digite ***$ mosquitto_sub -t "dw/demo"***
- Em uma outra janela do terminal, é possível usar o comando mosquitto_pub para conectar-se ao broker local e, em seguida, publicar uma mensagem em um tópico, sendo assim digite ***$ mosquitto_pub -t "dw/demo-m "hello world!"***

Após estes passos, na janela em que a flag *mosquitto_sub -t "dw/demo"* foi executada, irá aparecer na tela a mensagens hello world o que garante que o teste foi finalizado corretamente.

5.2 AMBIENTE DE TESTES

Será utilizado um computador (sistema operacional Linux Ubuntu, processador i5 e 8GB memória) para efetuar os disparos de publicadores/assinantes e também um raspberry pi 4 sendo o servidor qual o broker vai estar instanciado, as definições do Raspberry Pi 4 estão na seção a seguir.

5.2.1 Especificações do Raspberry Pi 4

Em um simples levantamento, a página oficial do Raspberry Pi traz que o dispostivo tem como padrão sistema operacional Raspian, baseado em Debian e está incorporado com um kernel linux (kernel versão 5.10), suas definições de hardware estão detalhadas na tabela a seguir. (16)

Tabela 1 – Especificações do Raspberry Pi 4 modelo B

Processador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memória	2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
Placa WIFI	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless
Placa de rede	Gigabit Ethernet até 1.000 Mb/s
Bluetooth	5.0, BLE
Portas USB	2 USB 3.0; 2 USB 2.0.
Audio	4-pole stereo audio and composite video port
Vídeo	2 × micro-HDMI ports
Gráficos	OpenGL ES 3.1, Vulkan 1.0
Pinos	40 pin GPIO header
Armazenamento	Micro-SD card slot for loading operating system and data storage
Voltagem	5V DC via USB-Connector (min. 3A*), 5V DC via GPIO header(min. 3A*)

Fonte: Pi Foundation (16)

5.2.2 Softwares de análise

Foi definido o Apache JMeter como ferramenta auxiliar durante as cargas de publicações e assinaturas para o estresse do Raspberry Pi 4, e justamente por ser uma ferramenta que realiza testes em recursos estáticos ou dinâmicos em sistemas de computação, o Apache JMeter é um software de código aberto e está disponível na página jmeter.apache.org.(2)

Durante o monitoramento do dispositivo a ser explorado será utilizado o RPi-Monitor qual permite o acompanhamento em tempo real do consumo dos componentes a serem analisados e garante também um log dos resultados obtidos, o software permite a exibição em gráfico de CPU, temperatura, memória, disco, etc. (19)

Por fim será utilizado o Wireshark para acompanhar o trafêgo da rede durante os testes. (23) Esta seção está sujeita a alterações pelos motivos de uma melhor demonstração gráfica dos dados estáticos após a execução dos os testes.

5.3 EXPERIMENTO

Em síntese os testes serão separados em três partes: um publicador para muitos assinantes, muitos publicadores para um assinante e por fim, muitos publicadores para muitos

assinantes, considerando esta última o maior desafio do projeto, em razão da própria construção do cenário muitos para muitos, ainda não visto em outros projetos. O design experimental ainda não foi definido, porém seram feitos diversos teste buscando encontrar o que melhor se adequará em cada experimento.

6 CRONOGRAMA

Atividades	Jul	Ago	Set	Out	Dez	Jan	Fev	Mar	Abr
Familiarização com a *protocolo MQTT*	**X**	**X**							
Análise literaria da rede IoT			**X**						
Desenvolvimento bibliográfico			**X**	**X**					
Elaboração e organização do experimento				**X**	**X**				
Aplicação do experimento					**X**	**X**			
Análise dos resultados							**X**		
Escrita da monografia								**X**	**X**

Tabela 2 – Cronograma previsto para as atividades propostas.

REFERÊNCIAS

1 ABES, Associação Brasileira de Empresas de Software. Mercado Brasileiro de Software: Panorama e Tendências 2021. In: 1. ed. São Paulo, SP, Brasil: ABES, 2021. p. 40. Disponível em: <https://abessoftware.com.br/dados-do-setor/>.

2 APACHE, JMeter™. **What can I do with it?** [S.l.]: The Apache Software foundation. Disponível em: <https://jmeter.apache.org/>.

3 ASHTON, Kevin. That 'Internet of Things' Thing. **RFID Journal**, jun. 2009. Disponível em: <https://www.rfidjournal.com/that-internet-of-things-thing>.

4 BASSI, Alessandro et al. **Enabling Things to Talk**: Designing IoT solutions with the IoT Architectural Reference Model. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2013. p. 349. ISBN 978-3-642-40403-0. DOI: 10.1007/978-3-642-40403-0.

5 BHATTACHARJEE, Sravani. **Pratical Industrial Internet of Things Security**: A practitioner's guide to securing connected industries. Birmingham, Englan, UK: Packt Publishing Ltd, jul. 2018. p. 324. ISBN 9781788832687. Disponível em: <https://www.packtpub.com/product/practical-industrial-internet-of-things-security/9781788832687>.

6 COMMUNITY, XMPP. **An Overview of XMPP**. [S.l.]: XMPP WebPage, 2002. Disponível em: <https://xmpp.org/about/technology-overview.html>.

7 DI MARTINO, B. et al. Internet of things reference architectures, security and interoperability: A survey. **Internet of Things**, Dipartimento di Ingegneria, Università della Campania Luigi Vanvitelli, Aversa (CE), Via Roma 29, Italy, v. 1-2, p. 99–112, 2018. ISSN 2542-6605. DOI: https://doi.org/10.1016/j.iot.2018.08.008. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2542660518300428>.

8 GCP, Google Cloud Platform. **Práticas e padrões híbridos e de várias nuvens**. [S.l.: s.n.], 2021. Disponível em: <https://cloud.google.com/architecture/hybrid-and-multi-cloud-patterns-and-practices?hl=pt-br>.

9 GÖTZ, Christian. **MQTT 101**: How to Get Started with the lightweight IoT Protocol. [S.l.]: Eclipse Foundation, 2014. Disponível em: <https://www.eclipse.org/community/eclipse_newsletter/2014/october/article2.php>.

10 GROUP, ITU-T Study. **New ITU standards define the Internet of Things and provide the blueprints for its development.** [S.l.: s.n.], 2012. Disponível em: <http://newslog.itu.int/archives/245>.

11 IBM, International Business Machines Corporation; EUROTECH. **MQTT specification version3.1.** [S.l.: s.n.], 2010. Disponível em: <http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html>.

12 MANYIKA, James et al. The Internet of Things: Mapping the value beyond the hype. In: [s.l.]: McKinsey Global Institute, jun. 2015. Disponível em: <https://apo.org.au/node/55490>.

13 MQTT. **FAQ**. [S.l.: s.n.]. Disponível em: <https://mqtt.org/faq/>.

14 MUENCHEN, Jean. **Uma proposta de detecção de incêndio utilizando protocolo MQTT para aplicações IOT.** Santa Maria, RS, Brasil: Universidade Federal de Santa Maria, jun. 2018. Disponível em: <http://repositorio.ufsm.br/handle/1/15799>.

15 PET, Sistemas de informação. **Computação em Nuvem**. [S.l.]: Universidade Federal de Santa Maria-UFSM, 2020. Disponível em: <https://www.ufsm.br/pet/sistemas-de-informacao/2020/09/15/computacao-em-nuvem/>.

16 PI FOUNDATION, Raspberry. **Raspberry Pi 4 Tech Specs**. [S.l.]: Raspberry Pi Org. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/>.

17 PROJECT, European lighthouse integrated. **Internet of Things**: Architecture. [S.l.]: 7 th framework programme, ago. 2019. Disponível em: <https://cordis.europa.eu/project/id/257521>.

18 RODRIGUES, Gabriel Augusto Veiga. **Análise de desempenho do beaglebone black utilizando o protocolo MQTT**. Ponta Grossa, PR, Brasil: Universidade Tecnológica Federal do Paraná, jun. 2019. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/12031>.

19 RPI-EXPERIENCES. **RPi-Monitor documentation**. [S.l.: s.n.]. Disponível em: <https://rpi-experiences.blogspot.com/>.

20 SILVA, Leandro Jamir. **Internet Das Coisas**. Palhoça, SC, Brasil: Universidade do Sul de Santa Catarina[UNISUL]., 2017. Disponível em: <https://repositorio.animaeducacao.com.br/handle/ANIMA/11220>.

21 SURESH, P. et al. A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In: 2014 International Conference on Science Engineering and Management Research (ICSEMR). [S.l.: s.n.], nov. 2014. p. 1–8. DOI: 10.1109/ICSEMR.2014.7043637.

22 TORRES, Andrei B. B.; ROCHA, Atslands R.; SOUZA, José Neuman de. Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo. In: XV Workshop em Desempenho de Sistemas Computacionais e de Comunicação. Porto Alegre, RS, Brasil: Sociedade Brasileira da Computação - SBC, 2016. (15), p. 2804–2815. DOI: https://doi.org/10.5753/wperformance.2016.9727.

23 WIRESHARK. **Learn Wireshark**. [S.l.]: the Wireshark Foundation. Disponível em: <https://www.wireshark.org/>.

24 YUAN, Michael. Conhecendo o MQTT: Por que o MQTT é um dos melhores protocolos de rede para a Internet das Coisas? **Article IBM developer**, out. 2017. Disponível em: <https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot/>.