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

CLEITON PICCINI

UMA ANÁLISE COMPARATIVA ENTRE OS PROTOCOLOS OPC-UA E MQTT EM UMA APLICAÇÃO DE IIOT

CHAPECÓ
2022

CLEITON PICCINI

UMA ANÁLISE COMPARATIVA ENTRE OS PROTOCOLOS OPC-UA E MQTT EM UMA APLICAÇÃO DE IIOT

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 Aurélio Spohn

CHAPECÓ
2022

Bibliotecas da Universidade Federal da Fronteira Sul - UFFS

Piccini, Cleiton
Uma análise comparativa entre os protocolos OPC-UA e MQTT em uma aplicação de IIoT / Cleiton Piccini. -- 2022.
35 f.:il.

Orientador: Dr. Marco Aurélio Spohn

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal da Fronteira Sul, Curso de Bacharelado em Ciência da Computação, Chapecó,SC, 2022.

1. Industrial Internet of Things. 2. Internet of Things. 3. MQTT. 4. OPC UA. I. Spohn, Marco Aurélio, orient. II. Universidade Federal da Fronteira Sul. III. Título.

Elaborada pelo sistema de Geração Automática de Ficha de Identificação da Obra pela UFFS com os dados fornecidos pelo(a) autor(a).

CLEITON PICCINI

UMA ANÁLISE COMPARATIVA ENTRE OS PROTOCOLOS OPC-UA E MQTT EM UMA APLICAÇÃO DE IIOT

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 Aurélio Spohn

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em: 15/8/2022.

BANCA AVALIADORA

Prof. Dr. Marco Aurélio Spohn – UFFS

Prof. Dr. Braulio Adriano de Mello – UFFS

Prof. Me. Adriano Sanick Padilha – UFFS

RESUMO

Com a crescente busca por aquisição de dados e supervisão de aplicações industriais, surgem cada vez mais soluções distintas para a troca de dados entre dispositivos. Um exemplo disto são os Industrial Internet of Things (IIOT), dispositivos industriais que visam o controle e troca de dados através de protocolos de rede. Devido a grande maioria destes equipamentos possuírem recursos computacionais limitados e buscarem maior eficiência energética, a escolha do protocolo de rede, dentre os inúmeros existentes, é primordial para a eficiência nas aplicações IIOT. Este trabalho busca efetuar um comparativo entre dois destes protocolos, tendo como objetivo descobrir qual protocolo possui maior eficácia. Os protocolos analisados são o Message Queuing Telemetry Transport (MQTT) e Open Platform Communications Unified Architecture (OPC UA) . O comparativo dos protocolos se dará através da análise de dados coletados a partir de métricas específicas de aplicações simuladas em diferentes cenários de desempenho.

Palavras-chave: Industrial Internet of Things. Internet of Things. MQTT. OPC UA.

ABSTRACT

With the growing search for data and monitoring of industrial applications, more and more differentiated solutions for exchanging data between devices. An example is the Industrial Internet of Things (IIOT), industrial devices that aim to control and exchange data through network protocols. Due to the vast majority of these resources needed for solutions and seeking greater energy efficiency, a network protocol, among the available resources, is essential for an application in IIOT applications. This work protocol seeks one of two of these, the objective of which is to find out which one has greater capacity. The protocols analyzed are the Message Queuing Telemetry Transport (MQTT) and Open Platform Communications Unified Architecture (OPC UA). The comparison of protocols will be analyzed through the analysis of simulated data from measurements analyzed in different performance scenarios.

Keywords: Industrial Internet of Things. Internet of Things. MQTT. OPC UA.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Os objetos do cotidiano estão cada vez mais dotados de tecnologia embarcada. Dispositivos simples como lâmpadas ou mais complexos como climatizadores, atualmente são capazes de reunir e transmitir diversos dados, além de serem controlados remotamente. Estes objetos fazem parte de um conceito chamado de Internet das Coisas (Internet of Things, IOT).

Estas novas capacidades e tecnologias embarcadas estendem-se aos dispositivos industriais de menor porte, como sensores e atuadores. Fisicamente pequenos, porém capazes de transmitir inúmeros dados pertinentes aos processos industriais, ou até mesmo manobrados de maneira remota, propriedades estas obtidas através dos protocolos de redes de comunicação. Estes equipamentos da industria com atributos semelhantes aos IOT, são denominados como Internet das Coisas Industriais (Industrial Internet of Things, IIOT).

A Indústria 4.0 é uma tendência atual na automação industrial, muitas vezes referida como a quarta revolução industrial. Seu principal objetivo é modernizar a forma de fabricação e produção, facilitando a integração de dispositivos e melhorando a comunicação entre todos os dispositivos no chão de fábrica. Componentes de diferentes fabricantes precisam se comunicar em uma linguagem comum usando protocolos de comunicação padronizados (PROFANTER et al., 2019).

A quarta revolução industrial tem por objetivo modernizar os métodos de produção para maior produtividade e menores desperdícios, através de plantas industriais automatizadas. Os dispositivos IIOT e controladores industriais têm um papel imprescindível para se atingir estes resultados. Equipamentos como sensores, atuadores, câmeras e controladores requerem a comunicação com Sistemas de Supervisão e Aquisição de Dados (Supervisory Control And Data Acquisition, SCADA) ou até mesmo aplicações na nuvem se torna cada vez mais frequente e necessária.

1.1 TEMA

Este trabalho tem como proposta analisar e comparar o desempenho de dois protocolos de comunicação implementados em inúmeros dispositivos IOT e IIOT. Os respectivos protocolos analisados serão a Plataforma de Comunicação Aberta de Arquitetura Unificada (Open Platform Communications Unified Architecture, OPC UA) e o Transporte de Telemetria de Enfileiramento de Mensagens (Message Queuing Telemetry Transport, MQTT). Ambos fortemente difundidos e muito utilizados em sistemas de automação por possuírem suporte ao Protocolo de Controle de Transmissão (Transmission Control Protocol, TCP) e o conceito Publicação/Assinatura (Publish/Subscribe) onde os Servidores ou Brokers podem publicar dados e os clientes podem se inscrever para os mesmos.

A análise visa efetuar a comparação dos recursos oferecidos por cada um dos protocolos, simulando diferentes cenários de aplicações.

1.2 PROBLEMATIZAÇÃO

Os dispositivos IOT e IIOT em grande maioria são equipamentos com baixo poder de processamento e recursos computacionais limitados, além de alguns dispositivos possuírem em sua essência a busca por menor tamanho físico e baixo consumo energético. Para isto, comparar e avaliar os protocolos MQTT e OPC UA tem como objetivo descobrir qual protocolo possui uma maior eficiência em diferentes estados tais como sistema ocioso, alta carga da Unidade de Processamento Central (Central Process Unit, CPU), alto tráfego da rede e alta alocação da memória. Conhecer o desempenho dos mesmos em diferentes cenários é de suma importância para a tomada de decisões na modelagem de aplicações ou no desenvolvimento de novos dispositivos IOT e IIOT.

À medida que os dispositivos IOT se expandem para uma infinidade de aplicações através da crescente minimização de hardware e a grande disponibilidade de sensores versáteis e "objetos inteligentes", muitos protocolos em potencial estão surgindo para comunicações Máquina a Máquina (Machine to Machine, M2M). A partir disso, a questão de qual protocolo utilizar para a IOT torna-se um tema de grande interesse. Devido à natureza remota e a necessidade de rede sem fio de objetos inteligentes, os sistemas IOT devem ser capazes de lidar com conexões potencialmente não confiáveis, intermitentes e de baixa largura de banda para sua rede de acesso (CHEN; KUNZ, 2016).

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos e seções. O capítulo 3 descreve a estrutura, os conceitos e elementos básicos para o funcionamento do MQTT e OPC UA. No capítulo 4 estão listados todos os trabalhos relacionados ao tema que contribuíram para a inspiração e elaboração deste projeto. No capítulo 5 são indicadas as métricas, a metodologia utilizada e a organização geral do trabalho. Os resultados do experimento estão situados no capítulo 6.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver uma aplicação com software livre, utilizando os protocolos MQTT e OPC UA para uma troca de dados simulada, focando na arquitetura IIOT. A aplicação tem por objetivo comparar o desempenho dos protocolos de comunicação analisando diferentes cenários simulados.

2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um sistema para simular aplicações IIOT, que se comuniquem utilizando MQTT e OPC UA;
- Através da aplicação devem ser coletados os dados de desempenho individual dos protocolos em diferentes estados do sistema;
- Avaliar os dados provenientes dos testes para definir qual protocolo obteve o melhor desempenho na simulação.

2.3 JUSTIFICATIVA

Para a modelagem de aplicações para automação industrial ou automação residencial existem inúmeros fatores relevantes a serem levados em consideração para elaboração deste tipo de aplicação. A definição do protocolo de comunicação que intermediará a troca de dados entre os sensores, atuadores ou qualquer outro dispositivo responsável pela coleta de dados do processo é de suma importância.

Para elaboração de aplicações consistentes e confiáveis, a troca de dados entre periféricos deve ser íntegra, e possuir tempo de resposta compatível com as necessidades da aplicação. Para isto, o protocolo escolhido deve atender as necessidades do processo ao qual se deseja automatizar. Este trabalho visa analisar o desempenho entre dois protocolos fortemente difundidos neste segmento, MQTT e OPC UA, para um eventual auxílio na definição de qual protocolo se utilizará em um projeto de automação, ou desenvolvimento de um dispositivo IOT ou IIOT. Pois o trabalho irá elencar as características de ambos, e qual deles possui o melhor desempenho na troca de dados.

3 REFERENCIAL TEÓRICO

3.1 MQTT

O MQTT é um protocolo de comunicação desenvolvido pela IBM e atualmente um padrão aberto da Organization for the Advancement of Structured Information Standards (OASIS). Foi projetado para utilizar mensagens leves, com a proposta de atender aplicações em redes não confiáveis ou com conectividade intermitente. Destaca-se pela capacidade de troca de dados em tempo real com aplicações na nuvem e pelo suporte ao monitoramento remoto.

É um protocolo da camada de aplicação, executado sobre o TCP e o Protocolo da Internet (Internet Protocol, IP). Utiliza o modelo Publish/Subscribe intermediado por um Servidor denominado de Broker. Os dados publicados são agrupados de forma hierárquica e organizados em tópicos, sendo possível que vários dispositivos distintos assinem ou publiquem no mesmo tópico. Segundo (AL-MASRI et al., 2020), esse modelo de comunicação é composto por um Broker, ou seja um servidor, e os clientes estabelecem uma conexão com ele a qualquer momento. Os clientes enviam mensagens por meio do Broker, processo conhecido como publisher. Em seguida, o Broker filtra essas mensagens recebidas e as distribui aos clientes interessados em recebê-las, procedimento denominado subscribe (modelo publicação-assinatura centralizado apresentado na figura 1).

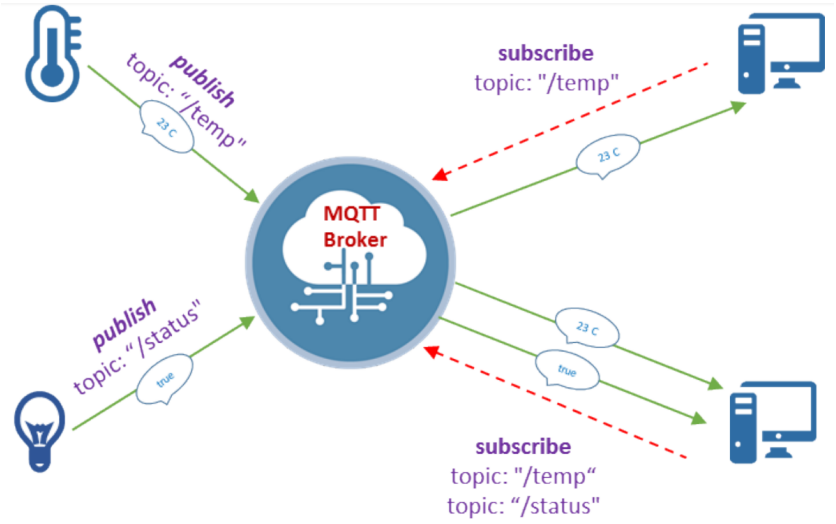

Figura 1 – Modelo de Broker MQTT centralizado.
Fonte: (AL-MASRI et al., 2020)

As mensagens podem possuir 3 níveis de Qualidade de Serviço (Quality of Service ,QoS) sendo eles:

- Nível 0 - No máximo uma vez:
  - Apenas uma tentativa. A mensagem é enviada apenas uma única vez, o destinatário não transmite resposta, o remetente não efetua nenhuma nova tentativa.

- Nível 1 - Pelo menos uma vez:
  - Garante que uma mensagem chegue ao receptor pelo menos uma vez. A mensagem é reenviada várias vezes, até que o receptor retorne uma mensagem ao remetente confirmando o recebimento.
- Nível 2 - Exatamente uma vez:
  - Garante que uma mensagem chegue ao destinatário exatamente uma vez sem duplicação. O emissor transmite e recebe a confirmação do receptor de que ele armazenou a mensagem. Se o emissor não receber uma confirmação, a mensagem será enviada novamente até que uma confirmação seja recebida.

O MQTT possui um tamanho máximo de carga útil (payload) de 256MB. Em quesitos de segurança dos dados, o MQTT suporta autenticação via nome de usuário e senha, ou usando uma infraestrutura de criptografia de chave privada Public Key Infrastructure (PKI).

3.2 OPC UA

O OPC UA surgiu em 1996 como um novo conceito para a automação industrial. Seu principal objetivo é a padronização da interação de controladores e dispositivos de campo. Possui como intuito isolar aplicações externas de detalhes dos equipamentos de automação e fornecer interfaces padronizadas para a troca de dados. Suas primeiras versões eram derivadas de tecnologias exclusivas Microsoft. Atualmente é um padrão independente de plataformas e é padronizado pela International Electrotechnical Commission (IEC), tendo como norma regulamentadora a IEC 62541-5. Segundo (PROFANTER et al., 2019), o OPC UA é um protocolo de comunicação máquina a máquina orientado a serviços, usado principalmente em automação industrial. Seus principais objetivos são fornecer comunicação multiplataforma usando um modelo de informação que descreve os dados transferidos.

OPC UA é um protocolo da camada de aplicação, que pode ser executado sobre o TCP ou User Datagram Protocol (UDP). Utiliza o modelo cliente-servidor, o qual se baseia em uma arquitetura onde o servidor é detentor dos recursos e serviços da rede. Os clientes por sua vez solicitam a ele os atributos que desejam utilizar, porém o cliente não compartilha nenhum de seus recursos com o servidor, exceto a troca de dados. Nesse modelo cliente-servidor, o OPC UA tem a capacidade de utilizar o Remote Procedure Call (RPC), funcionalidade esta que permite que clientes executem chamadas de procedimentos remotos ao servidor. Recentemente, o OPC UA apresentou uma especificação de Publicação-Assinatura semelhante ao MQTT, onde os servidores publicam os dados e clientes podem assiná-los.

O OPC UA não possui QoS nativo, porém é possível obtê-lo a partir de ferramentas de terceiros. O protocolo não oferece suporte para troca de dados com a Nuvem. Em quesitos de segurança dos dados, suporta autenticação via nome de usuário e senha, ou usando uma infraestrutura de criptografia de chave privada (PKI).

4 TRABALHOS RELACIONADOS

(PROFANTER et al., 2019) efetuou a comparação de recursos de diferentes protocolos utilizando as implementações de código aberto Open6254 para OPC UA e Paho MQTT para MQTT. A comparação consistiu em medir tempo de resposta, utilização da memória, processamento da CPU, mensagens por segundo e o Tempo de Ida e Volta (Round-Trip Time, RTT) nos seguintes estados da aplicação:

- Ocioso. Dentro de 30 segundos, um cliente espera por um período de tempo aleatório entre 0 e 3 segundos.
- Alta carga da CPU utilizando o software Stress12 (cria carga artificial da CPU).
- Alta carga da rede utilizando o software Ostinato11 (cria carga de rede artificial).

Para medir a taxa de transferência foi utilizado o método do eco (Echo), onde o servidor responde com os mesmos dados enviados pelo cliente. O tempo de resposta foi medido através da confirmação (Acknowledge, ACK), onde um cliente envia uma solicitação e aguarda uma mensagem de resposta. Solicitações foram feitas continuamente até que 5.000 solicitações fossem enviadas. Após atingir este número de solicitações, o tamanho da carga útil é dobrado, começando de 2 bytes e indo até 32.768 bytes.

Para o teste de caso de uso, foram iniciadas 500 instâncias de Servidor ao mesmo tempo em uma máquina. O Cliente efetua o envio simultâneo de pacotes para nós em execução no Servidor, os mesmos encaminham o pacote recebido para o próximo nó. Todo este processo foi repetido 100 vezes para obter a média dos resultados.

Os testes indicaram que o OPC UA não se correlaciona com a carga da CPU; no entanto, o MQTT sofre desaceleração com alta demanda da CPU. A implementação Open62541 OPC UA Pub/Sub é mais rápido para diferentes tamanhos de pacotes. O Paho MQTT, por sua vez, exige mais recursos da CPU para processar as mensagens, além de ser o que mais apresentou outliers. Os testes utilizaram duas máquinas com as seguintes características CPU Intel i7-8700K com 3,70 GHz e 64 GB de RAM.

(RADDATZ et al., 2020) criou uma aplicação utilizando Open62541 em linguagem de programação C para OPC UA e o Broker Mosquitto para o MQTT. O hardware utilizado para os testes consiste em um switch full duplex de 100 Mbps e dois Raspberry Pi 3 como dispositivos de campo. Foi efetuado medições para avaliar o desempenho do OPC UA e MQTT utilizando assinatura cliente/servidor e publicação/assinatura usando UDP multicast (UADP).

Para OPC UA Cliente/Servidor é avaliado o desempenho em relação ao tempo total de transmissão de dados, efetuando a medição do tempo necessário para a troca de dados entre Clientes e Servidor OPC UA. A medição tem início após a solicitação de assinatura do Cliente ser confirmada pelo Servidor, e termina quando o mesmo recebe uma mensagem de notificação do Servidor. No caso do padrão de Publicação/Assinatura OPC UA com UADP, é medido a

latência entre o assinante e o publicador usando UDP. A medição começa quando o publicador envia seus dados e termina após o assinante receber a mensagem publicada.

Para o MQTT o Publicador e os Assinantes assinam um tópico específico no Broker. Foi definida duas medidas de tempo no fluxo de comunicação. O primeiro é obtido diretamente após o publicador enviar uma mensagem ao Broker. A segunda medida de tempo é obtida quando o assinante recebe a mensagem encaminhada. O tempo médio de transmissão para cada canal de comunicação é apresentado na Figura 2.

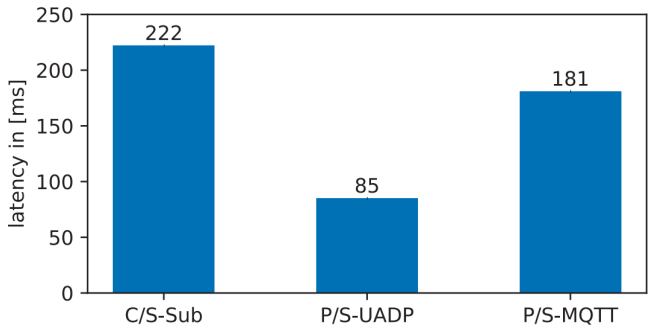

Figura 2 – Média das medições de latência.
Fonte: (RADDATZ et al., 2020)

O OPC UA utilizando o padrão Cliente/Servidor apresenta um atraso maior em relação às mensagens usando Publicação/Assinatura. A comunicação usando MQTT requer mais tempo para trocar uma mensagem do que a variante UADP, constatando que a comunicação baseada em UDP não requer esse tempo de processamento, pois não há intermediário entre eles.

A Figura 3 mostra o atraso médio do MQTT para cada parte da comunicação entre o Broker e o assinante. O tempo necessário para receber a mensagem publicada pelo assinante aumenta à medida que o tamanho da mensagem aumenta.

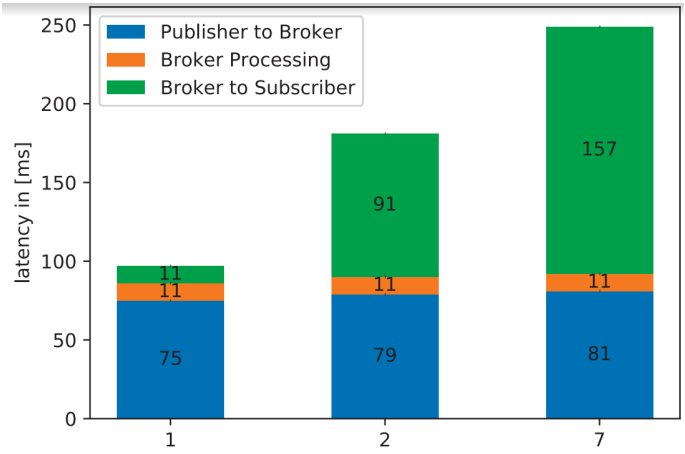

Figura 3 – Latência por número de variáveis transmitidas.
Fonte: (RADDATZ et al., 2020)

5 METODOLOGIA

5.1 IMPLEMENTAÇÃO

O presente estudo é do tipo experimental, e visa medir e comparar o desempenho dos protocolos MQTT e OPC UA. Para isto, três aplicações de simulação foram desenvolvidas, sendo estas, Servidor OPC UA, Clientes OPC UA e Publicadores/Assinantes MQTT. Como Broker MQTT foi utilizado a ferramenta Open Source *Broker Eclipse Mosquitto*. Com a finalidade de coletar informações de desempenho de ambos protocolos citados, a simulação tem seus dados armazenados e analisados para o comparativo dos dois protocolos. Os métodos e técnicas utilizados para a simulação, foram adaptados a partir dos ensaios e estudos realizados por (PROFANTER et al., 2019). Os dados adicionais são obtidos através das biblioteca *XlsxWriter 3.0.3 em linguagem Python 3.7.3*, utilizada para persistir os dados coletados em arquivos do tipo *.xlsx* e *psutil 5.9.1 em linguagem Python 3.7.3*, responsável por obter o processamento da CPU. O código fonte das aplicações pode ser encontrado no repositório online *https://github.com/cleitonPiccini/TCC*.

5.1.1 Hardware

Na execução das aplicações utilizou-se dois microcomputadores *RaspBerry Pi Model B+*. Os dispositivos possuem processador 1.2GHz Quad-Core ARM Cortex-A53 64Bit, memória RAM 1GB LPDDR2, Porta Ethernet LAN 10/100 BaseT Ethernet socket. Equipamento pode ser visualisado na figura 4.

Para executar a aplicação do Servidor OPC UA e Broker MQTT o equipamento recebeu um cartão Micro SD Samsung 64Gb Classe 10 para armazenamento de dados, para os Clientes OPC UA e Publicadores/Assinantes MQTT foi utilizado um cartão Micro SD SanDisk 32Gb Classe 10. Ambos os dispositivos utilizam o sistema operacional Raspberry Pi OS (Legacy) 32Bit.

Os dois dispositivos são interligados via Porta Ethernet utilizando um cabo de rede 6SL3060-4AJ20-0AA0 Siemens.

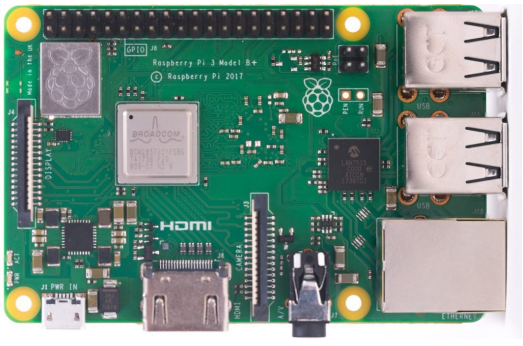

Figura 4 – Visão superior do Raspberry Pi 3 Model B+
Disponpivel em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>

5.1.2 Servidor OPC UA

O Servidor foi desenvolvido utilizando *FreeOPC-UA 0.90.6 em linguagem Python 3.7.3*, a biblioteca em questão detém as funções necessárias para desenvolver a aplicação essencial para a troca de informações com os Clientes OPC UA.

A aplicação é iniciada via terminal Linux, efetuando a execução do arquivo *ServerOPC.py*, a definição de endereço IP, porta de comunicação e nome do servidor é feita através do arquivo de configuração *ConfigOPC.txt*. Após obter os dados de configuração, as variáveis OPC UA são definidas, os métodos de inserção são criados e o Servidor está pronto para a troca de dados com os Clientes. As etapas de execução da aplicação são apresentadas na figura 5.

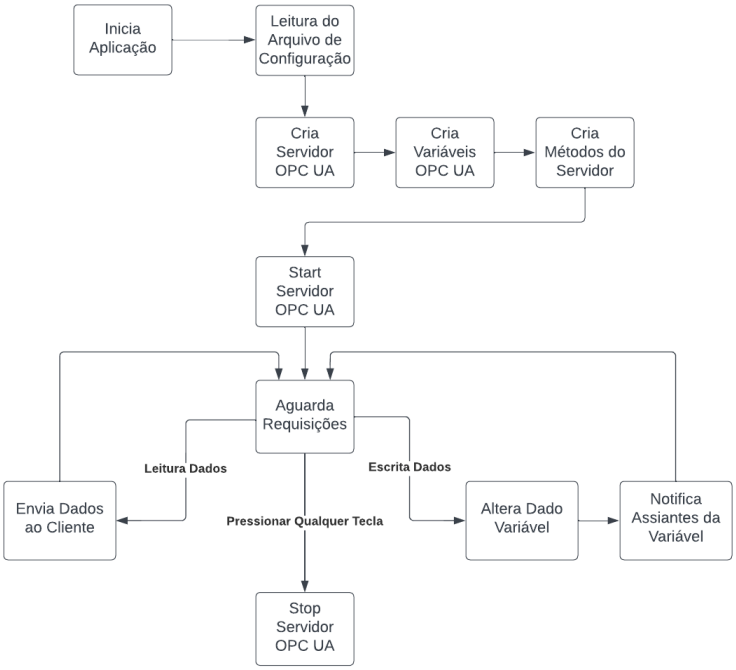

Figura 5 – Fluxo de execução do Servidor OPC UA.

5.1.3 Clientes OPC UA

Os Clientes OPC UA foram desenvolvidos utilizando *FreeOPC-UA 0.90.6 em linguagem Python 3.7.3*, a biblioteca em questão detém as funções necessárias para desenvolver a aplicação que acessa as informações contidas no Servidor OPC UA. A mesma não possui a troca de dados utilizando UDP, apenas TCP.

A aplicação é iniciada via terminal Linux, através da execução do arquivo *StartClientOPC.py*, as informações necessárias para a conexão com o Servidor OPC UA como nome, endereço IP, porta de comunicação, número de clientes gerados através de chamadas multi-threads e o tipo de ensaio que será executado são obtidas do arquivo de configuração *ConfigOPC.txt*.

Após iniciada a simulação gera os Clientes, cria a conexão, estabelece o vínculo com as respectivas variáveis do servidor, gera o arquivo para armazenar os dados coletados e efetua a

assinatura das variáveis utilizadas no ensaio. A aplicação possui dois tipos de ensaios o Echo e o Acknowledgement (Ack).

Echo é o ensaio do tempo de transferência, onde o Cliente envia uma mensagem ao Servidor e o mesmo responde com os mesmos dados enviados pelo Cliente. Acknowledgement é o método do tempo de resposta, onde o Cliente envia uma mensagem ao Servidor e aguarda uma mensagem de confirmação do mesmo. Em ambos os ensaios a carga do processador, uso de memória RAM e o tempo cronometrado entre a troca de informações são registrados a cada mensagem, as mesmas iniciam no tamanho mínimo em bytes, até que atinjam o número máximo de envios, após, efetua-se o cálculo de média destes valores, cálculo de desvio padrão dos registros de tempo e dobra-se o tamanho da carga útil, chegando até o tamanho máximo em bytes. Valores de mínimo e máximo definidos no arquivo de configuração *ConfigOPC.txt*. As etapas de execução da aplicação são apresentadas na figura 6.

Figura 6 – Fluxo de execução dos Clientes OPC UA.

5.1.4 Publicadores/Assinantes MQTT

Os Publicadores e Assinantes MQTT foram desenvolvidos utilizando *Paho MQTT 1.6.1 em linguagem Python 3.7.3*, a biblioteca em questão detém as funções necessárias para desenvolver a aplicação que acessa as informações contidas no *Broker Eclipse Mosquitto*.

A aplicação é iniciada via terminal Linux, através da execução do arquivo *StartMQTT.py*, as informações necessárias para a conexão com o *Broker Eclipse Mosquitto* como endereço IP, porta de comunicação, número de Publicadores/Assinantes gerados através de chamadas multi-threads e o tipo de ensaio que será executado são obtidas do arquivo de configuração

ConfigMQTT.txt. Antes de iniciar os Publicadores/Assinantes o *Broker Mosquitto* precisa estar rodando com as mesmas configurações de porta de comunicação contidas em *ConfigMQTT.txt*.

Após iniciada a simulação gera os Publicadores/Assinantes, cria a conexão, estabelece os tópicos no *Broker Eclipse Mosquitto*, gera o arquivo para armazenar os dados coletados e efetua a assinatura dos tópicos utilizados no ensaio. A aplicação possui dois tipos de ensaios o Echo e o Acknowledgement.

Echo é o ensaio do tempo de transferência, onde o Publicador envia uma mensagem ao *Broker Eclipse Mosquitto* e o mesmo responde com os mesmos dados enviados pelo Publicador. Acknowledgement é o método do tempo de resposta, onde o Publicador envia uma mensagem ao *Broker Eclipse Mosquitto* e aguarda uma mensagem de confirmação do mesmo. Em ambos os ensaios a carga do processador, uso de memória RAM e o tempo cronometrado entre a troca de informações são registrados a cada mensagem, as mesmas iniciam no tamanho mínimo em bytes, até que atinjam o número máximo de envios, após, efetua-se o cálculo de média destes valores, cálculo de desvio padrão dos registros de tempo e dobra-se o tamanho da carga útil, chegando até o tamanho máximo em bytes. Valores de mínimo e máximo definidos no arquivo de configuração *ConfigMQTT.txt*. As mensagens utilizam o nível 0 de QoS. As etapas de execução da aplicação são apresentadas na figura 7.

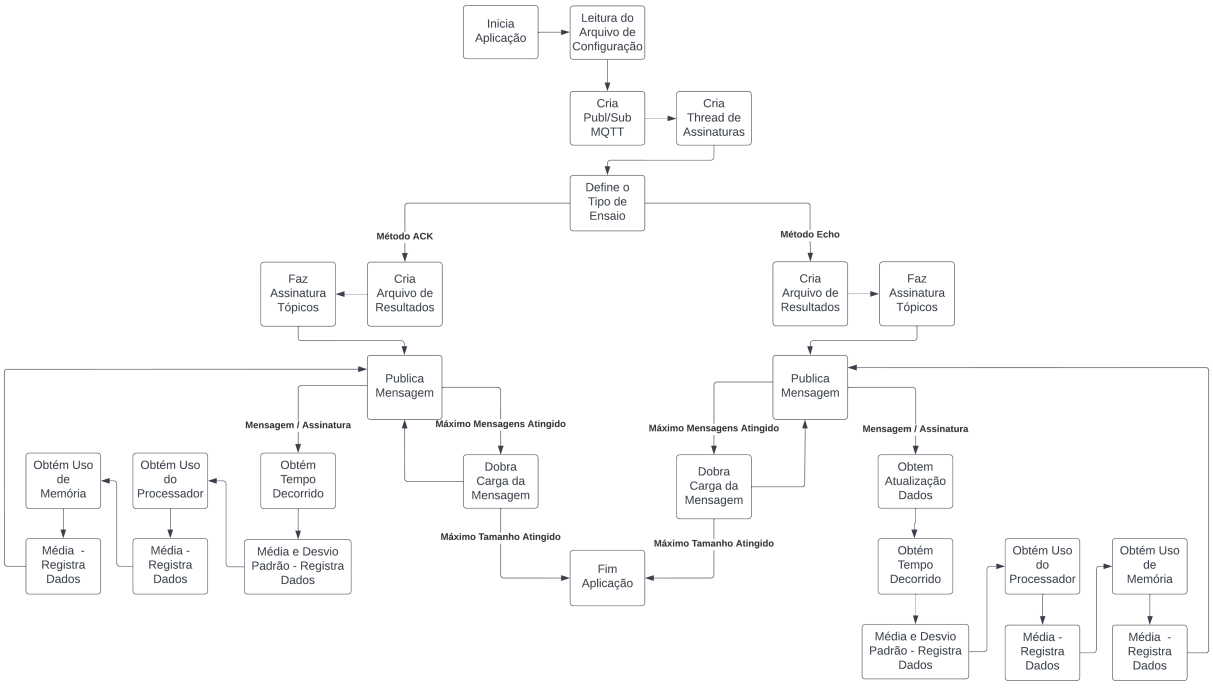

Figura 7 – Fluxo de execução dos Publicadores e Assinantes MQTT.

6 RESULTADOS

6.1 OPC UA

Uma série de modelos foram analisados para compor os resultados de desempenho do OPC UA, sendo estes executados de forma isolada, onde apenas a aplicação responsável pelos testes e as funções mínimas do sistema operacional estavam em operação. Cada teste foi executado por inúmeros Clientes conectados ao Servidor OPC UA. Durante os testes algumas dificuldades foram encontradas, devido às características do hardware utilizado, o dispositivo possui baixa capacidade de processamento e pouca memória de trabalho, a fim de contornar estes obstáculos, vários ensaios foram feitos até que fosse descoberto o número de Clientes que o equipamento suporta executar.

Para a simulação, o hardware suporta a execução simultânea de 40 Clientes. Uma quantidade maior pode ser utilizada, porém a troca de dados torna-se lenta ocasionando falhas de "timeout", gerando assim "exceptions" que finalizam a aplicação antes que ela complete todo o seu ciclo de execução. A quantidade máxima de mensagens enviadas com a mesma carga, e o limite máximo de dados, foram definidas através da análise do trabalho de (PROFANTER et al., 2019).

A biblioteca *FreeOPC-UA 0.90.6 em linguagem Python 3.7.3* apresentou inconsistências em sua implantação, a IEC 62541-5 define uma série de funções as quais devem ser implementadas, algumas destas não foram encontradas na documentação da biblioteca, como por exemplo, a função de *CallBack PosWrite()*, responsável por notificações de uma variável quando seus dados sofrem alteração. A biblioteca contempla em sua implementação o controle autônomo dos processos de escrita, leitura e assinatura de variáveis, deste modo, o desenvolvedor não precisa elaborar códigos multi-thread para troca de dados e notificações do Servidor, a mesma os cria.

Os resultados demonstram que a aplicação OPC UA possui tempos de atrasos superiores a três vezes o MQTT. A partir da simulação observou-se que para cinco Clientes o tempo de atraso na troca de mensagens no método Ack é menor que a metade do que para vinte, porém para o método Echo os valores se assemelham. O Servidor OPC UA apresentou ineficiência na troca de dados e baixo gerenciamento aos múltiplos acessos.

Os resultados podem ser observados nas figuras 8 e 9, resultados utilizando os métodos Ack e Echo com 20 Clientes, e figuras 10 e 11, resultados utilizando os métodos Ack e Echo com 5 Clientes. Os mesmos também podem ser observados na tabela 1.

6.2 MQTT

Através da análise de inúmeros modelos foram compostos os resultados de desempenho do MQTT, sendo estes executados de forma isolada, onde apenas a aplicação responsável pelos

testes e as funções mínimas do sistema operacional estavam em operação. Cada teste foi executado por inúmeros usuários conectados ao *Broker Mosquitto*. Durante os testes algumas dificuldades foram encontradas, devido às características do hardware utilizado, as mesmas dificuldades e soluções citadas no item 6.1.

Para a simulação o hardware suporta a execução simultânea de 60 Publicadores/Assinantes, devido às restrições da aplicação OPC UA o número foi restringido a apenas 20, para que os testes fossem imparciais. A quantidade máxima de mensagens enviadas com a mesma carga, e o limite máximo de dados, foram definidos através da análise do trabalho de (PROFANTER et al., 2019).

A biblioteca *Paho MQTT 1.6.1 em linguagem Python 3.7.3* demonstrou-se muito consistente. A mesma necessita que o desenvolvedor crie um processo responsável apenas pelo controle das notificações de Assinaturas dos Tópicos existentes no Broker, as quais são mantidas em processamento através da função de loop já implementada na biblioteca, deste modo a aplicação possui dois processos distintos, um para as Assinaturas e outro para as Publicações.

A aplicação apresentou resultados acima do esperado. A simulação mostrou-se linear ao executar um número pequeno ou grande de usuários. O Broker Mosquitto demonstrou-se eficiente na troca de dados, e excelente no gerenciamento aos múltiplos acessos. Os resultados podem ser observados nas figuras 12 e 13, resultados utilizando métodos Ack e Echo com 20 Publicadores/Assinantes, e figuras 14 e 15, utilizando métodos Ack e Echo com 5 Publicadores/Assinantes.

6.3 OPC UA E MQTT

Foram realizados testes com os dois protocolos sendo executados simultaneamente no mesmo conjunto de hardware, a fim de observar o comportamento das aplicações trabalhando ao mesmo tempo. Para isto foi elaborado os seguintes ensaios:

- Método Echo com 20 Clientes OPC UA, 20 Publicadores/Assinantes MQTT, 5000 mensagens iniciando em 2 bytes e finalizando com 32768 bytes;
- Método Ack com 20 Clientes OPC UA, 20 Publicadores/Assinantes MQTT, 5000 mensagens iniciando em 2 bytes e finalizando com 32768 bytes.
- Método Echo com 5 Clientes OPC UA, 5 Publicadores/Assinantes MQTT, 500 mensagens iniciando em 2 bytes e finalizando com 32768 bytes;
- Método Ack com 5 Clientes OPC UA, 5 Publicadores/Assinantes MQTT, 500 mensagens iniciando em 2 bytes e finalizando com 32768 bytes.

Os resultados a partir da utilização do método Ack são apresentados na tabela 3 e os resultados no método Echo na tabela 4. Pode-se observar o restante dos dados nas figuras 16,

17, 18 e 19. Nas imagens é possível analisar que nos testes com 15 Clientes, o OPC UA tem uma diminuição do atraso nas interações com maior tamanho de dado, isto deve-se ao fato da aplicação MQTT encerar seu ensaio antes, devido ao menor tempo de atraso, desta forma a simulação OPC UA passa a ser processada de forma isolada, gerando assim uma melhora na troca de dados.

OPC UA - Método Ack	**20 Clientes**	**5000 Mensagens**	**5 Clientes**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	33,50705696		15,6477392	
Tamanho do Dado - Bytes	32768		32768	
OPC UA - Método Echo	**20 Clientes**	**5000 Mensagens**	**5 Clientes**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	42,50556972		39,503516	
Tamanho do Dado - Bytes	32768		32768	

Tabela 1 – Comparativo resultados OPC UA método Ack e Echo.

MQTT - Método Ack	**20 Pub/Sub**	**5000 Mensagens**	**5 Pub/Sub**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	1,9507039		1,3578156	
Tamanho do Dado - Bytes	32768		32768	
MQTT - Método Echo	**20 Pub/Sub**	**5000 Mensagens**	**5 Pub/Sub**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	1,99812323		1,9859696	
Tamanho do Dado - Bytes	32768		32768	

Tabela 2 – Comparativo resultados MQTT método Ack e Echo.

OPC UA	**15 Clientes**	**5000 Mensagens**	**5 Clientes**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	32,65904596		16,5273636	
Tamanho do Dado - Bytes	2		32768	
MQTT	**15 Clientes**	**5000 Mensagens**	**5 Clientes**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	1,895803107		1,3578156	
Tamanho do Dado - Bytes	32768		32768	

Tabela 3 – Comparativo resultados OPC UA e MQTT no método Ack.

OPC UA	**15 Clientes**	**5000 Mensagens**	**5 Clientes**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	123,5863247		40,39764	
Tamanho do Dado - Bytes	256		32768	
MQTT	**15 Clientes**	**5000 Mensagens**	**5 Clientes**	**500 Mensagens**
Maior Tempo de Atraso - Milissegundos	6,067060893		1,3715542	
Tamanho do Dado - Bytes	32768		32768	

Tabela 4 – Comparativo resultados OPC UA e MQTT no método Echo.

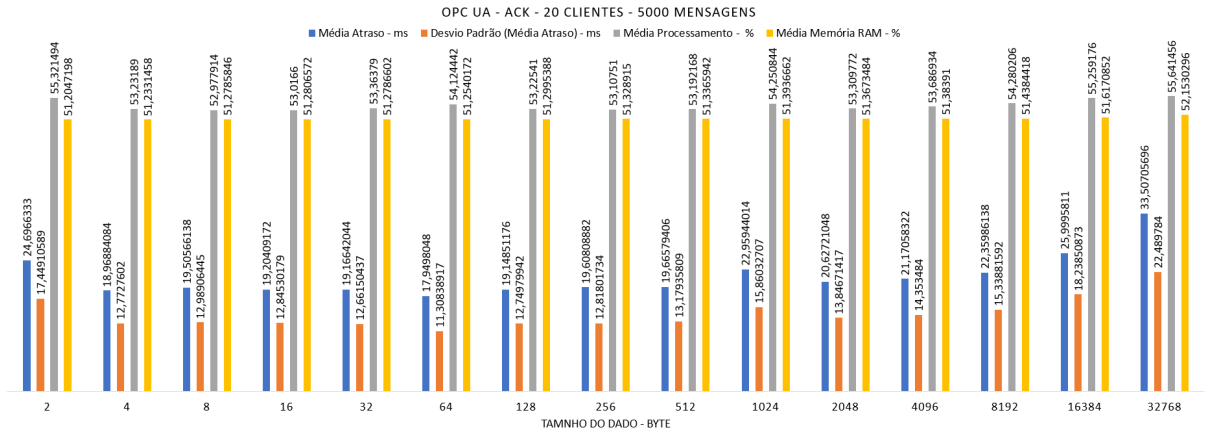

Figura 8 – Protocolo OPC UA utilizando o método Ack com 20 Clientes simultâneos.

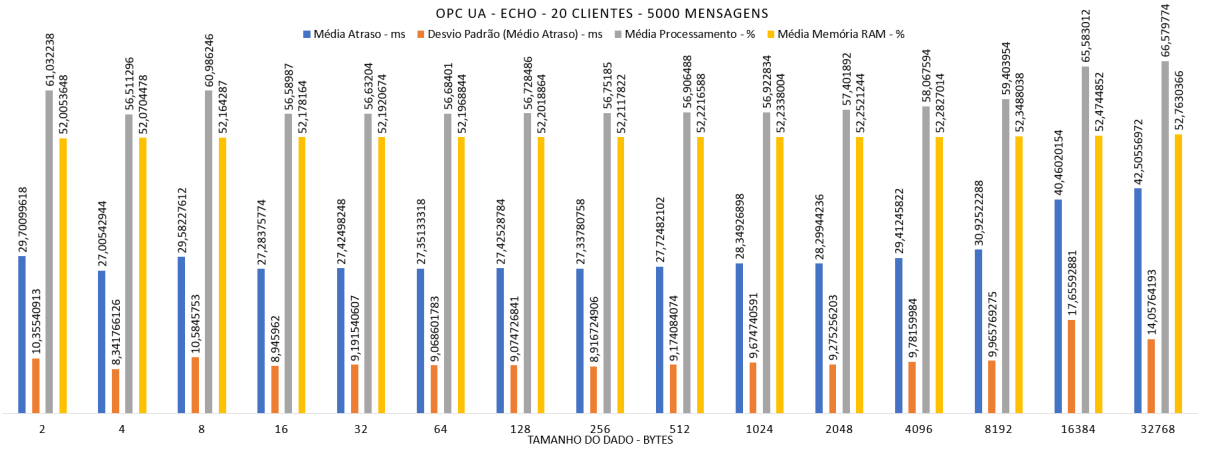

Figura 9 – Protocolo OPC UA utilizando o método Echo com 20 Clientes simultâneos.

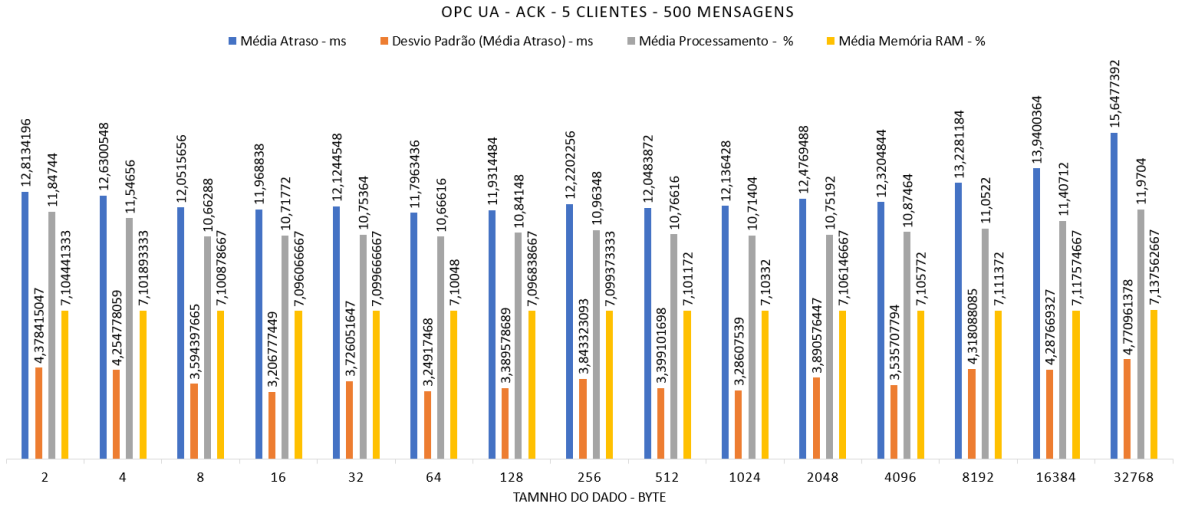

Figura 10 – Protocolo OPC UA utilizando o método Ack com 5 Clientes simultâneos.

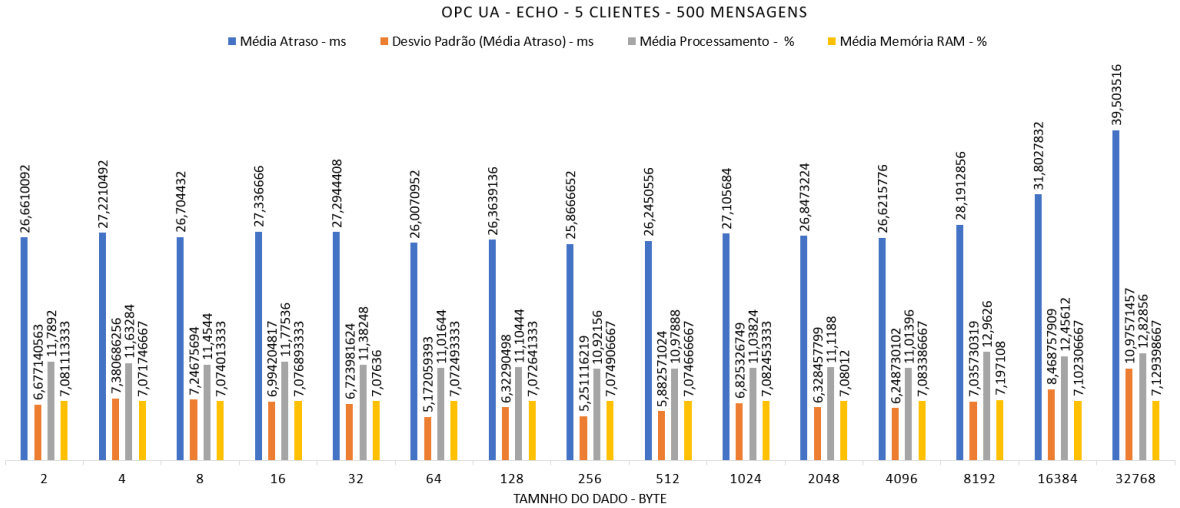

Figura 11 – Protocolo OPC UA utilizando o método Echo com 5 Clientes simultâneos.

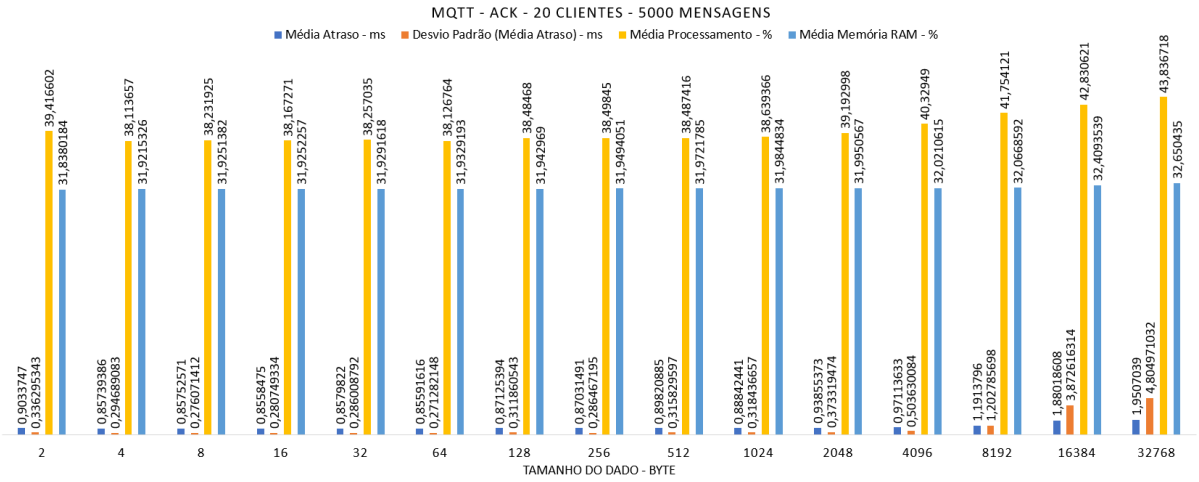

Figura 12 – Protocolo MQTT utilizando o método Ack com 20 Publicadores/Assinantes simultâneos.

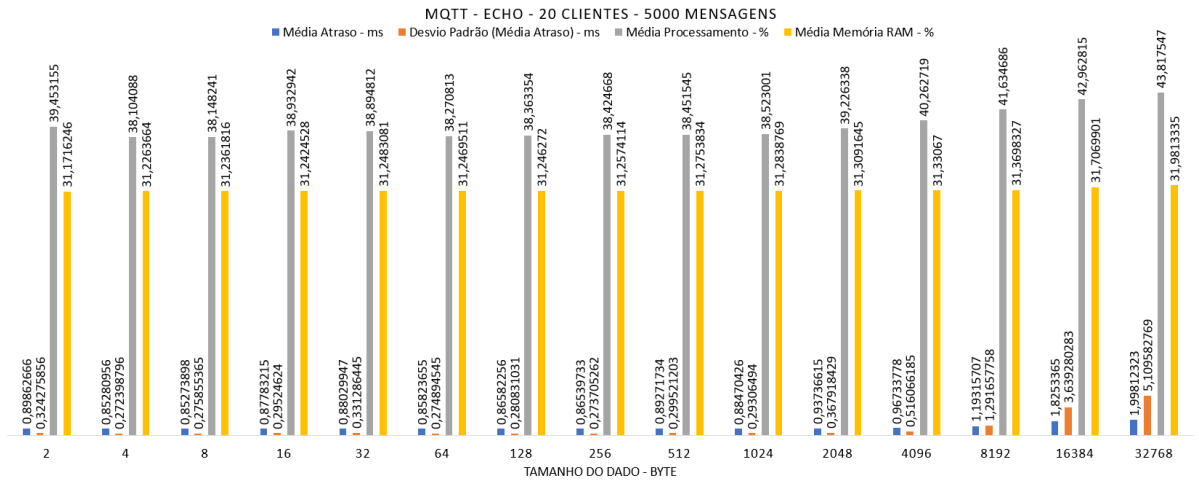

Figura 13 – Protocolo MQTT utilizando o método Echo com 20 Publicadores/Assinantes simultâneos.

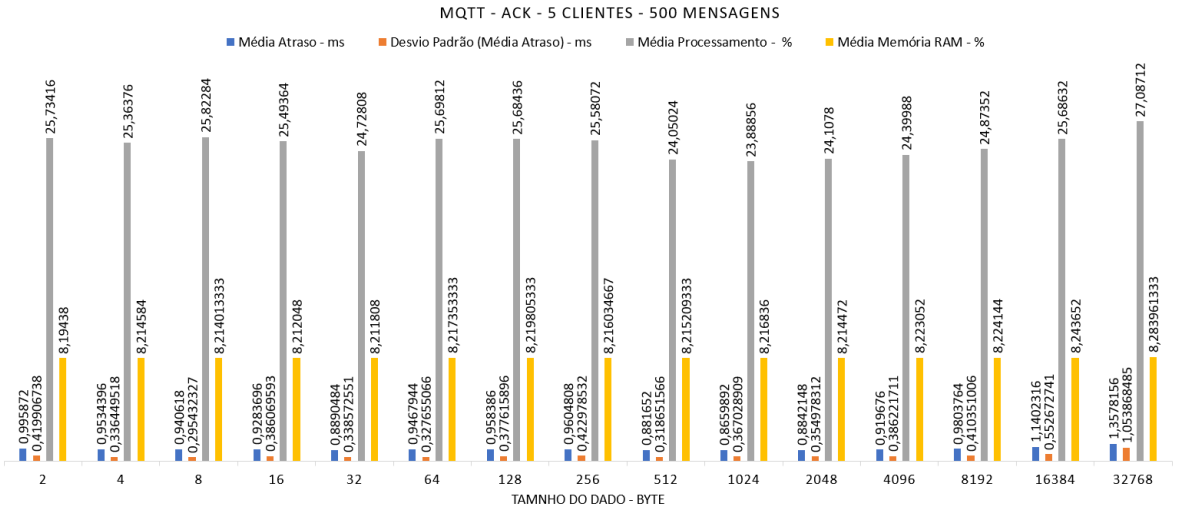

Figura 14 – Protocolo MQTT utilizando o método Ack com 20 Publicadores/Assinantes simultâneos.

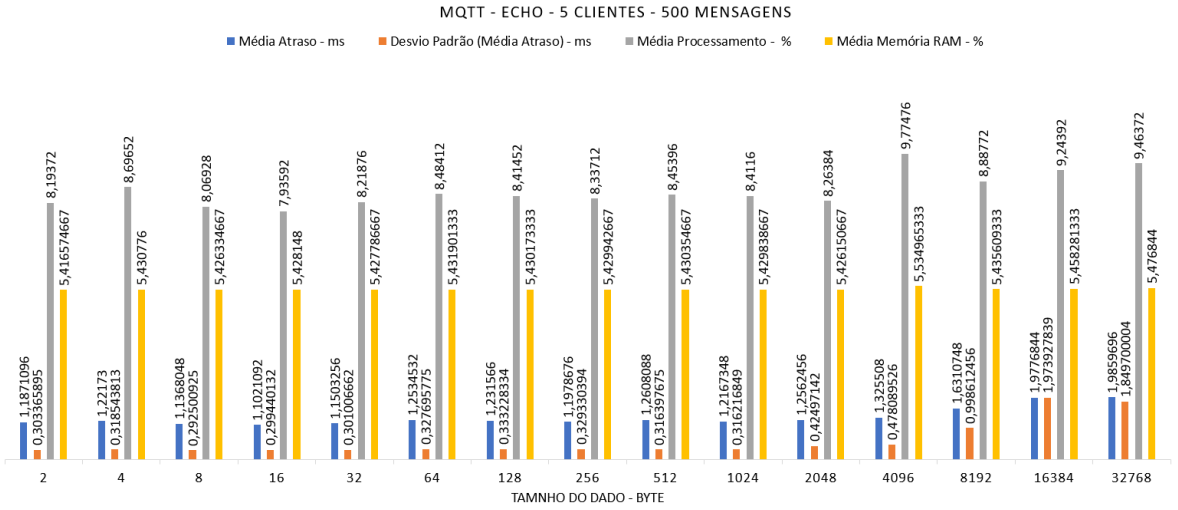

Figura 15 – Protocolo MQTT utilizando o método Echo com 20 Publicadores/Assinantes simultâneos.

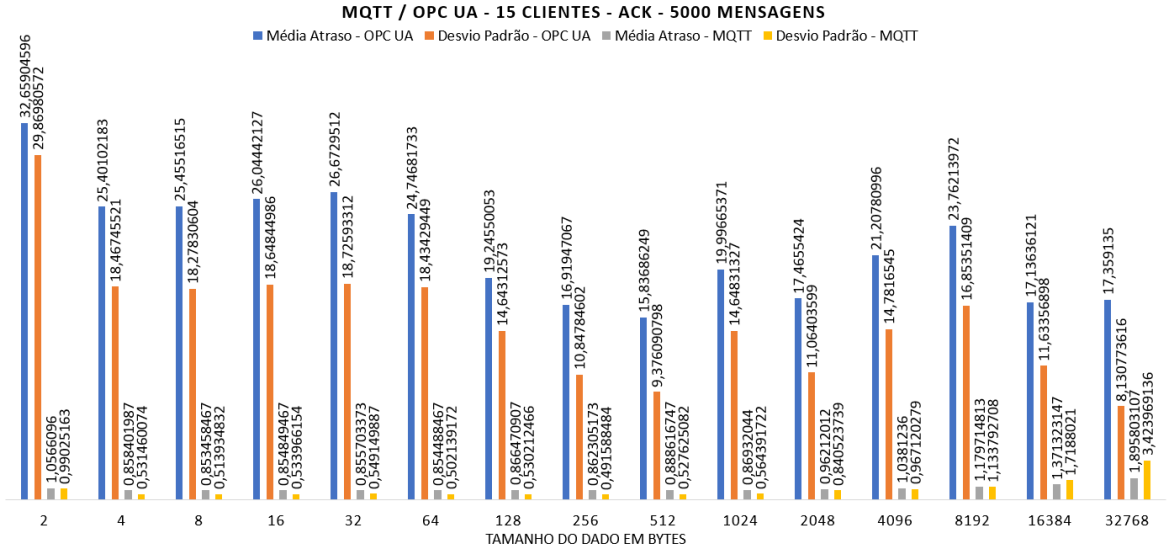

Figura 16 – Protocolos MQTT e OPC UA utilizando método Ack com 15 Clientes e 15 Publicadores/Assinantes.

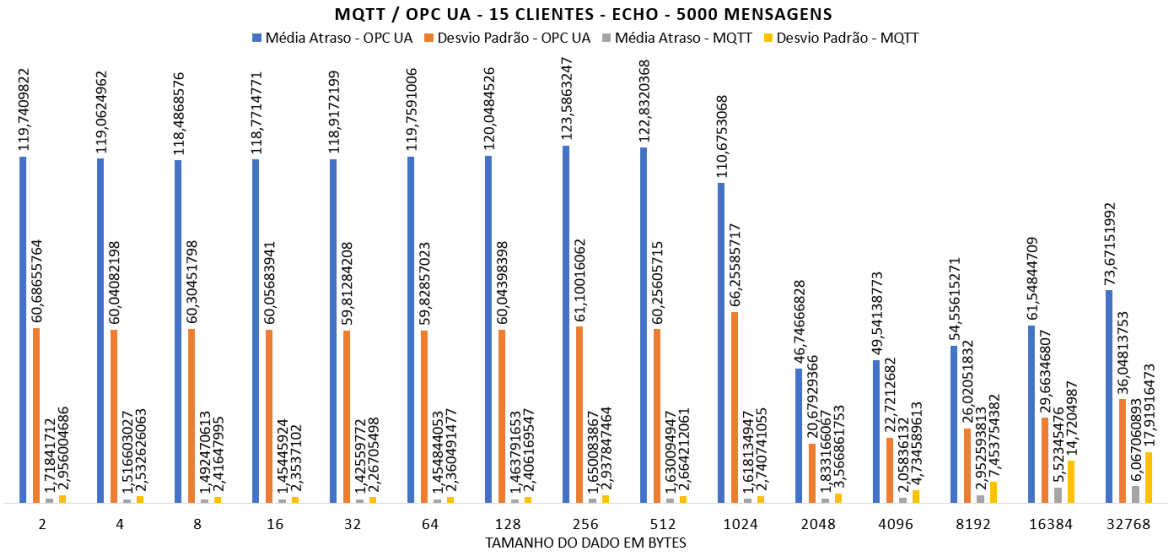

Figura 17 – Protocolos MQTT e OPC UA utilizando método Echo com 15 Clientes e 15 Publicadores/Assinantes.

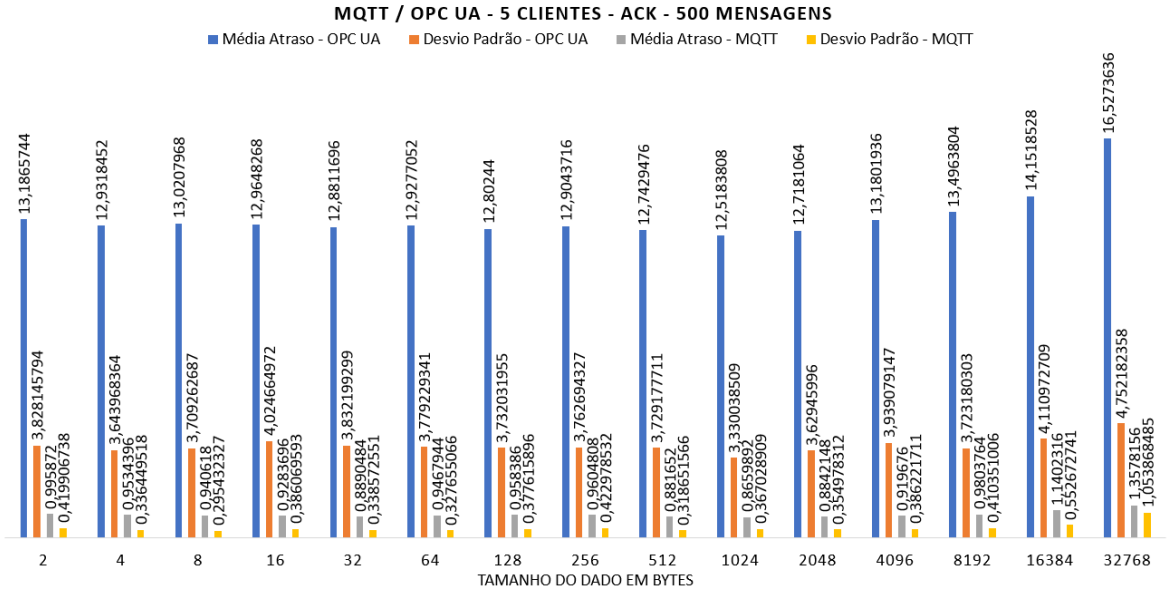

Figura 18 – Protocolos MQTT e OPC UA utilizando método Ack com 5 Clientes e 5 Publicadores/Assinantes.

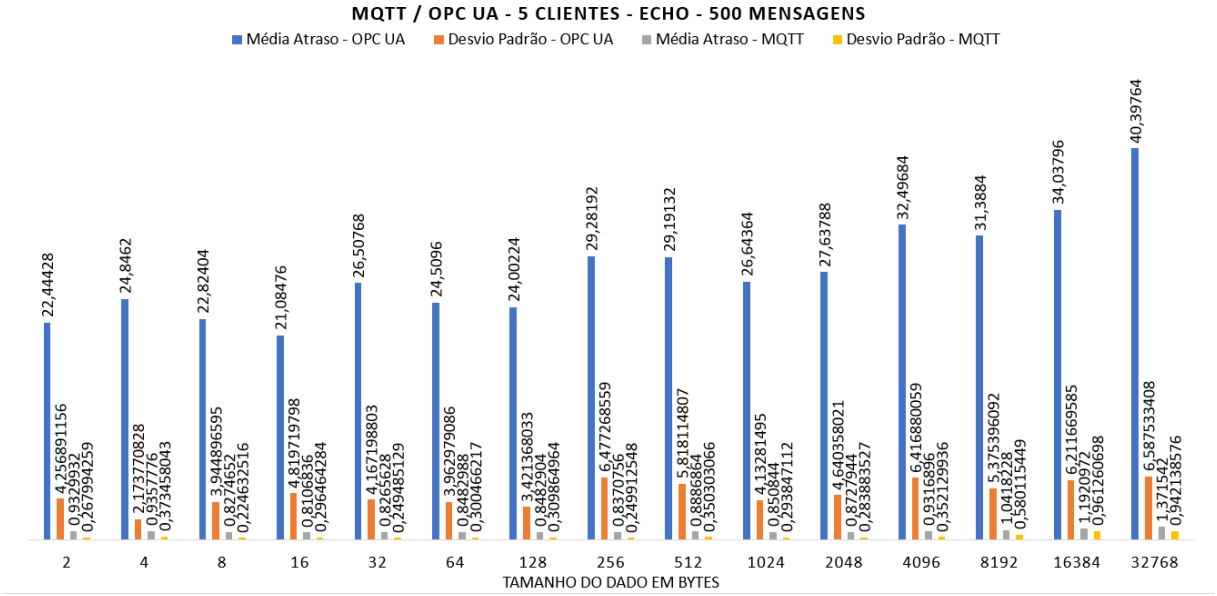

Figura 19 – Protocolos MQTT e OPC UA utilizando método Echo com 5 Clientes e 5 Publicadores/Assinantes.

7 CONCLUSÃO

Os objetivos propostos foram desenvolver um sistema para simular aplicações IIOT que se comuniquem utilizando MQTT e OPC UA, coletar seus dados de desempenho através da simulação e comparar os resultados dos protocolos em questão, os mesmos foram alcançados.

Após comparar os dados provenientes da aplicação desenvolvida para este trabalho, conclui-se que, para os presentes cenários e estados estabelecidos para os testes da simulação, o protocolo MQTT obteve um tempo de atraso nas trocas de mensagens menor que o OPC UA. Deste modo, levando em consideração apenas a referência de tempo entre troca de mensagens, pode-se afirmar que ele possui um melhor desempenho que neste quesito, pois no uso de memória RAM e carga do processador ambos protocolos possuem resultados semelhantes.

O MQTT obteve menor tempo de atraso na troca de mensagens em todos os modelos de ensaio, fossem eles utilizando poucas conexões e mensagens, ou mesmo várias, e até mesmo em execução conjunta com o OPC UA.

O uso da memória RAM permanece praticamente inalterado em todos os testes com ambos os protocolos, independente do tamanho das mensagens, mostrando-se assim que tal grandeza sofreu pouca interferência nas simulações. A carga da CPU por sua vez sofre interferências ao longo dos testes, mostrando uma leve relação entre ela e o tempo de atraso na troca de mensagens, a mesma possui valores semelhantes em ambos os protocolos.

Observando os métodos já utilizados e novas bibliotecas para ambos os protocolos que estão surgindo, um novo conjunto de testes pode ser desenvolvido como uma proposta de trabalho futuro. Bem como a utilização de uma biblioteca para OPC UA, a qual possua suporte para troca de mensagens utilizando UDP ou até mesmo utilizar ensaios englobando sistemas SCADA.

REFERÊNCIAS

CHEN, Yuang; KUNZ, Thomas. Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network, 2016. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7496622>.

AL-MASRI, Eyhab et al. Investigating Messaging Protocols for the Internet of Things (IoT), 2020. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9090208>.

PROFANTER, Stefan et al. OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols, 2019. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8755050>.

RADDATZ, Hannes et al. Evaluation and Extension of OPC UA Publish/Subscribe MQTT Binding, 2020. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9274696>.

SILVA, Daniel et al. A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA, 2021. Disponível em: <https://www.mdpi.com/2076-3417/11/11/4879>.