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

MARCELO ELIAS KNOB

AVALIAÇÃO DA FUNCIONALIDADE DE ASSINATURAS COMPARTILHADAS NO MQTT V5.0

CHAPECÓ
2022

MARCELO ELIAS KNOB

AVALIAÇÃO DA FUNCIONALIDADE DE ASSINATURAS COMPARTILHADAS NO MQTT V5.0

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

Knob, Marcelo Elias
Avaliação da funcionalidade de assinaturas compartilhadas no MQTT v5.0 / Marcelo Elias Knob. -- 2022.
39 f.:il.

Orientador: Doutor 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. IoT. 2. M2M. 3. MQTT. 4. 5.0. 5. Assinaturas compartilhadas. 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).

MARCELO ELIAS KNOB

AVALIAÇÃO DA FUNCIONALIDADE DE ASSINATURAS COMPARTILHADAS NO MQTT V5.0

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: 10/8/2022.

BANCA AVALIADORA

Prof. Dr. Marco Aurélio Spohn – UFFS

Prof. Me. Adriano Sanick Padilha – UFFS

Prof. Dr. Braulio Adriano de Mello – UFFS

RESUMO

O número de dispositivos conectados à internet cresce continuamente e rapidamente formando a Internet das Coisas (*Internet of Things* - IoT). Tendo em vista que todos esses dispositivos comunicam entre si e com outros serviços, a preocupação de garantir um sistema tolerante à falhas, seguro e otimizado vem à tona. Para que essa comunicação aconteça, são utilizados protocolos máquina a máquina (M2M - *Machine To Machine*), o qual se destaca o MQTT (*Message Queuing Telemetry Transport*). O MQTT é um protocolo de transporte de mensagens cliente e servidor que funciona no modelo publicação/assinatura e é executado sobre a rede TCP/IP. Além disso, o protocolo é simples, leve, aberto e projetado para ser fácil de implementar. Assim sendo, o MQTT é, devido às suas características, o protocolo mais utilizado e difundido em IoT e em comunicações M2M. Em 2019, a versão 5.0 foi lançada acompanhada de diversas funcionalidades, dentre elas a de assinaturas compartilhadas. Essa funcionalidade promete balancear a carga entre os assinantes de determinado tópico, entretanto, por ser uma funcionalidade recente, pouco ainda do seu potencial foi explorado. Portanto, o presente trabalho propõe uma avaliação dos aspectos comportamentais e de desempenho do MQTT v5.0 com foco na funcionalidade de assinaturas compartilhadas.

Palavras-chave: IoT. M2M. MQTT. 5.0. Assinaturas compartilhadas.

ABSTRACT

The number of devices connected to the internet grows continuously and rapidly forming the Internet of Things (IoT). Given that all these devices communicate with each other and with other services, the concern to ensure a fault-tolerant, secure and optimized system comes up. For this communication to happen, machine-to-machine (M2M) protocols are used, in which MQTT (Message Queuing Telemetry Transport) stands out. MQTT is a client server message transport protocol that works on the publish/subscribe model and runs over the TCP/IP network. Furthermore, the protocol is light weight, open, simple and designed so as to be easy to implement. Therefore, MQTT is, due to its characteristics, it is the most used and widespread protocol in IoT and M2M communications. In 2019, version 5.0 was released accompanied by several features, including shared subscriptions. This functionality promises to balance the load between subscribers of certain topic, however, as it is a recent functionality, little of its potential has yet been explored. Therefore, the present work proposes an evaluation of the behavioral and performance aspects of MQTT v5.0 with a focus on the functionality of shared subcriptions.

Keywords: IoT. M2M. MQTT. 5.0. Shared subscription.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

A existência de dispositivos físicos, independente do seu tamanho, ou sensores que conseguem trocar informações, como relógio, pulseira, geladeira, via conexão à internet formam a IoT (9). A IoT permite a coleta de dados monitorados desses dispositivos em uma rede sem qualquer interação humana a fim de melhorar a nossa vida, negócios ou ambientes. A aplicação da IoT vai além da área de transporte, indústria, agricultura ou saúde, os espaços de aplicação são praticamente ilimitados. Ou seja, a IoT permite a comunicação de dispositivos interconectados para que consigam coletar, armazenar, processar, descrever e analisar dados para resolver uma variedade de problemas.

Atualmente, bilhões de dispositivos ou coisas inteligentes estão em ambientes IoT e o número de dispositivos interconectados cresce de forma contínua e rápida. Diante disso, a comunicação de dispositivos entre si e com outros serviços é dada de máquina para máquina (M2M), sendo o desempenho fortemente dependente dos protocolos de mensagens especiais projetados para essa comunicação em aplicações IoT. Assim, são necessários protocolos que garantam a segurança, eficiência e tolerância a falhas, sendo o MQTT, CoAP, AMQP e HTTP os quatro protocolos de mensagens amplamente aceitos e emergentes para sistemas IoT.

De acordo com o relatório **IoT & Edge Developer Survey Report 2021** (5), o MQTT lidera como o protocolo mais adotado em IoT e o servidor MQTT como a tecnologia mais comum usada para infraestrutura de mensagens, representando, em ambos os casos, 44% da preferência. Sendo assim, o MQTT é o protocolo mais difundido em sistemas M2M e IoT por ser leve, aberto, simples e fácil de ser implantado, apesar de não limitar-se a esses sistemas. Além disso, é um padrão OASIS e ISO aberto (ISO/IEC PRF 20922) para cliente e servidor no modelo de mensagens publicação/assinatura, é executado sobre a rede TCP/IP (*Transmission Control Protocol / Internet Protocol*) e permite conexões com autenticação SSL/TLS (*Secure Sockets Layer / Transport Layer Security*).

O MQTT é constituído por três componentes: um produtor/publicador (um cliente MQTT); um *broker* (um servidor MQTT); e um consumidor/assinante (um cliente MQTT). O cliente pode ser um aplicativo ou dispositivo que utiliza o protocolo MQTT e é responsável por criar e publicar as mensagens para o servidor, além de assinar e receber as mensagens aos quais está interessado. A mensagem do aplicativo são os dados transportados e carregam em si informações das suas propriedades. Já o servidor atua como intermediário entre os clientes assinantes e publicadores, fazendo o gerenciamento das mensagens, solicitações e conexões. Como o MQTT é um protocolo com comunicação bidirecional, isso auxilia no compartilhamento de dados, no gerenciamento e no controle de dispositivos.

O MQTT está disponível em duas versões, a 3.1.1, padronizada pela OASIS em 2014, e a 5.0, publicada em 2018 e padronizada em 2019. O lançamento da versão 5.0 foi acompanhada

pela introdução de diversas funcionalidades, dentro elas a de assinaturas compartilhadas. Essa funcionalidade promete o balanceamento da carga das mensagens entre os membros assinantes de um grupo, na qual somente um dos assinantes receberá a mensagem, de forma a diminuir o fluxo de mensagens e aliviar a carga dos assinantes e do *broker*.

1.2 PROBLEMÁTICA

Por meio da funcionalidade de assinaturas compartilhadas, a escalabilidade horizontal de um sistema pode ser ampliada de forma otimizada e eficiente, visto que cada mensagem é processada por apenas um cliente assinante. A escalabilidade horizontal permite o aumento do desempenho de um sistema pela introdução de máquinas clientes ou *brokers* para distribuir a carga. Dessa forma, o MQTT v5.0 aprimorou esse recurso com as assinaturas compartilhadas, sendo relevante em sistemas de grande escala.

Entretanto, para que essa funcionalidade exerça sua função, algoritmos de roteamento são utilizados pelo *broker* para decidir o destino das mensagens aos clientes assinantes de um grupo de assinaturas compartilhadas. Como esses algoritmos utilizam diferentes parâmetros para definir o cliente destino, resultados diferentes poderão ser obtidos. Por exemplo, o algoritmo *random* escolhe um dos clientes aleatoriamente entre os membros do grupo, enquanto que o algoritmo *round-robin* escolhe de cada vez um dos membros do grupo até todos serem designados, para então repetir o mesmo processo. Dessa forma, é presumível a ocorrência de resultados divergentes entre os algoritmos de roteamento.

Apesar da expectativa que a funcionalidade de assinaturas compartilhadas oferece, incertezas quanto ao seu desempenho podem ser observadas. Isso acontece em virtude tanto das divergência encontradas nos algoritmos de roteamento como da ausência de pesquisas que procuram avaliar o desempenho da funcionalidade. Isto posto, é relevante comparar o desempenho da mesma, a qual integra-se ao MQTT v5.0, com um sistema operando sem a funcionalidade, proporcionando uma avaliação comparativa entre ambas as diferentes situações de sistema.

A ferramenta *MQTTLoader*, citada posteriormente no capítulo 5 (Trabalhos Relacionados) do presente trabalho, possibilita a comparação anteriormente referida, já que suporta e consegue trabalhar com a versão 5.0 do MQTT. Logo, é uma ferramenta que auxiliará na análise do desempenho da funcionalidade de assinaturas compartilhadas e na suscetibilidade da escalabilidade horizontal do MQTT.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

Avaliar aspectos comportamentais e de desempenho do MQTT v5.0, com ênfase em suas características de escalabilidade horizontal.

2.2 OBJETIVOS ESPECÍFICOS

- Avaliar o serviço de assinaturas compartilhadas no MQTT v5.0;
- Aplicar a ferramenta *MQTTLoader* no MQTT v5.0 para analisar o desempenho do serviço de assinaturas compartilhadas.

3 JUSTIFICATIVA

A escalabilidade horizontal é uma da soluções mais utilizadas por empresas para atender grande quantidade de solicitações, enquanto que no MQTT isso é praticável aumentando a quantidade de clientes ou de *brokers*. Entretanto, na versão 3.1.1 do MQTT, todas as mensagens, solicitações, publicadas em determinado tópico são enviadas para todos os seus assinantes. Logo, isso pode ser um problema, visto o grande fluxo de mensagens ocorrendo entre o *broker* e os assinantes, gerando gargalo entre as partes e comprometendo o desempenho da aplicação.

Tendo em vista o problema de gargalo, uma aplicação que não necessite que todos os assinantes recebam as mesmas mensagens, mas apenas um deles, a utilização de escalabilidade horizontal acaba sendo ineficiente. Entretanto, no MQTT v5.0, o serviço de assinaturas compartilhadas foi introduzido prometendo resolver essa ineficiência, de forma que cada mensagem publicada é entregue a somente um assinante. Dessa forma, a carga do *broker* como dos assinantes seria aliviada e balanceada, visando o melhor desempenho da aplicação.

Além disso, como esse serviço não existia nativamente pelo *broker* na versão 3.1.1, a solução necessitava da intervenção do desenvolvedor em alterar o código fonte para implementá-lo. Assim, com a disponibilidade que a versão 5.0 fornece, gerenciar, analisar e processar dados de IoT se tornou uma tarefa muito mais fácil e simples.

Portanto, o serviço de assinaturas compartilhadas é uma solução para escalabilidade horizontal visando a melhoria no desempenho e eficiência de uma aplicação. Apesar disso, a exploração desse serviço ainda demanda de pesquisas para avaliar a sua suscetibilidade de implantação em sistemas reais. Logo, o presente trabalho objetiva a avaliação dos aspectos comportamentais e de desempenho do MQTT v5.0, com ênfase em seus aspectos de escalabilidade horizontal por meio da utilização da funcionalidade de assinaturas compartilhadas.

4 REFERENCIAL TEÓRICO

4.1 INTERNET OF THINGS

A IoT é caracterizada pela conectividade de qualquer dispositivo inteligente criador de dados à internet e viabiliza o gerenciamento desses dados para que, por último, sejam usufruídos por pessoas para tomarem decisões, independente do seu espaço de atuação, baseadas na análise dessas informações.

O modelo de referência de IoT, padronizado globalmente pela Cisco em 2014, define uma infraestrutura que inclui sete camadas e que podem ser visualizadas na figura 1. As camadas da figura 1 compreende: o nível 1, representado pelos dispositivos físicos; nível 2, pela conectividade; nível 3, pela computação em borda da rede, denominado de *Edge* ou *Fog*, onde ocorre o processamento inicial dos dados; nível 4 e 5, pela computação em nuvem, onde ocorre o armazenamento e a abstração dimensionada dos dados; e nível 6 e 7, onde ocorre a interpretação de informações e relatórios a fim de tomar uma decisão perspicaz, já que tomar as medidas apropriadas geralmente envolve pessoas e processos.

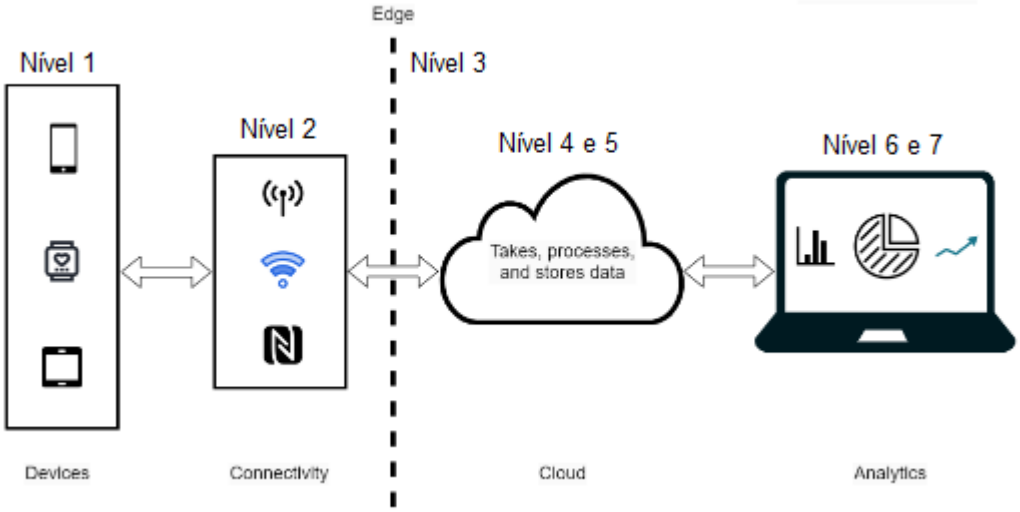

Figura 1 – Arquitetura geral de IoT reproduzido de Mishra; Kertesz (9).

4.2 MESSAGE QUEUE TRANSPORT TELEMETRY

Foi inventado por Dr. Andy Stanford-Clark da IBM e Arlen Nipper da Arcom em 1999 com o princípio de minimizar a largura de banda da rede e os requisitos de recursos do dispositivo com o propósito de garantir uma entrega confiável. Foi padronizado pela OASIS em 2014 e pela ISO em 2015, ambas organizações reguladoras de padrões. As características de funcionamento estão dadas nas sub sessões a seguir conforme a documentação **MQTT v3.1.1** (10).

4.2.1 Publicação/Assinatura

É o modelo de comunicação adotado pelo MQTT e que acontece entre cliente e servidor. Uma publicação compreende o nome e filtro de tópico e uma Qualidade de Serviço (*Quality of Service* - QoS) máxima, mas não permite o uso de curinga. Uma publicação está associada a uma única sessão. Já uma assinatura compreende um filtro de tópico e uma QoS máxima, além de estar associada a uma única sessão. Uma sessão pode conter mais de uma assinatura. Cada assinatura em uma sessão tem um filtro de tópico diferente. A finalização da assinatura dos tópicos ocorre enviando uma mensagem para o servidor.

4.2.2 Servidor

Um programa ou dispositivo que atua como intermediário entre clientes que publicam mensagens de aplicativo e clientes que fizeram assinaturas. O servidor pode:

- Aceitar conexões de rede de clientes;
- Aceitar mensagens de aplicativos publicadas por clientes;
- Processar solicitações de assinaturas e cancelamento de assinaturas de clientes;
- Encaminhar as mensagens do aplicativo que correspondem às assinaturas do cliente.

4.2.3 Cliente

Um programa ou dispositivo que usa MQTT. Um cliente sempre estabelece a conexão de rede com o servidor. O cliente pode:

- Publicar mensagens de aplicativos que possam interessar a outros clientes;
- Assinar para solicitar as mensagens de aplicativo que tem interesse em receber;
- Cancelar a assinatura para remover uma solicitação de mensagens do aplicativo;
- Desconectar-se do servidor.

4.2.4 Sessão

Uma interação com estado entre um cliente e um servidor. Algumas sessões duram tanto quanto a conexão de rede, outras podem abranger várias conexões de rede consecutivas entre um cliente e um servidor.

4.2.5 Tópico

O nome do tópico é abrangido pelo rótulo anexado a uma mensagem de aplicativo que corresponde às assinaturas conhecidas pelo servidor. O servidor envia uma cópia da mensagem do aplicativo para cada cliente que possui uma assinatura correspondente. Além disso, o tópico pode compreender um filtro, que é uma expressão contida em uma assinatura, para indicar interesse em um ou mais tópicos. Um filtro de tópico pode incluir caracteres curinga.

4.2.6 Qualidade de Serviço

O MQTT entrega mensagens de aplicativo de acordo com os níveis de QoS. O protocolo de entrega se preocupa exclusivamente com a entrega de uma mensagem de aplicativo de um único remetente para um único receptor. Quando o servidor está entregando uma mensagem de aplicação para mais de um cliente, cada cliente é tratado de forma independente. O nível de QoS usado para entregar uma mensagem de aplicativo de saída ao cliente pode ser diferente daquele da mensagem de aplicativo de entrada. Os três níveis de QoS estão descritos a seguir:

- QoS 0 (no máximo uma entrega): a mensagem é entregue de acordo com os recursos da rede subjacente. Nenhuma resposta é enviada pelo destinatário e nenhuma nova tentativa é realizada pelo remetente. A mensagem chega ao receptor uma vez ou não chega;
- QoS 1 (pelo menos uma entrega): garante que a mensagem chegue ao receptor pelo menos uma vez, permitindo mensagens duplicadas;
- QoS 2 (exatamente uma entrega): é a mais alta qualidade de serviço, para uso quando nem a perda nem a duplicação de mensagens são aceitáveis. Há uma sobrecarga maior associada a essa qualidade de serviço.

4.2.7 Segurança

A autenticação do cliente ao servidor é obtida pelo pacote de conexão que armazena campos de usuário e senha para serem aplicados na implantação ao sistema configurado. As implantações podem escolher como fazer uso desse conteúdo e fornecer soluções próprias ou externas de autenticação. Quanto a segurança do servidor, o MQTT recomenda altamente que as implementações de servidor que oferecem TLS devem usar a porta TCP 8883.

4.2.8 Versão 5.0

Conforme a documentação **MQTT v5.0** (11), essa versão adiciona um número significativo de novos recursos ao MQTT, mantendo grande parte do núcleo no lugar. Segundo Mileva et al. (8) alguns dos recursos introduzidos que se destacam, estão:

- Gerenciamento de sessão aprimorado: com os intervalos opcionais de expiração de sessão e mensagem;
- Restrições do servidor: um servidor pode definir um conjunto de recursos que não suporta;
- Restrições do cliente: um cliente pode definir um conjunto de recursos que não suporta;
- Alias de tópico: utilização de pequenos inteiros em vez de nomes de tópicos para reduzir o tamanho dos pacotes de controle;
- ID de assinatura: possibilita que o cliente determine qual assinatura ou assinaturas fizeram com que a mensagem fosse entregue;
- Assinaturas compartilhadas.

Assinaturas não compartilhadas

Uma assinatura não compartilhada é associada apenas à sessão MQTT que a criou. Cada assinatura inclui um filtro de tópico, indicando os tópicos para os quais as mensagens devem ser entregues nessa sessão e as opções de assinatura. O servidor é responsável por coletar as mensagens que correspondem ao filtro e transmiti-las na conexão MQTT da sessão se e quando essa conexão estiver ativa.

Se houver vários clientes, cada um com sua própria assinatura não compartilhada para o mesmo tópico, cada cliente obtém sua própria cópia das mensagens do aplicativo publicadas nesse tópico. Isso significa que as assinaturas não compartilhadas não podem ser usadas para balancear a carga das mensagens do aplicativo em vários clientes consumidores, pois nesses casos todas as mensagens são entregues a todos os clientes assinantes.

Assinaturas compartilhadas

Uma assinatura compartilhada pode ser associada a várias sessões MQTT de assinatura. Assim como uma assinatura não compartilhada, ela possui um filtro de tópicos e opções de assinatura; no entanto, uma publicação que corresponda ao seu filtro de tópico só é enviada para uma de suas sessões assinantes. As assinaturas compartilhadas são úteis quando vários clientes consumidores compartilham o processamento das publicações em paralelo.

5 TRABALHOS RELACIONADOS

5.1 MEDINDO O DESEMPENHO DE SERVIDORES DO MQTT V5.0 COM MQTTLOADER

Banno et al. (2) apresentam uma ferramenta de teste de carga para MQTT implementada na linguagem de programação Java chamada *MQTTLoader*. Essa ferramenta visa criar vários clientes e aplicar uma carga em um *broker* específico de acordo com as configurações fornecidas pelo usuário. Além disso, arquivos binários, código fonte e documentos estão disponíveis no GitHub (3).

A aplicação abrange diversos parâmetros, dentre eles: endereço do *broker*, versão do MQTT, número de publicadores e assinantes, habilitar assinaturas compartilhadas, endereço do servidor *Network Time Protocol* (NTP), entre outros. A figura 2 mostra uma visão geral da ferramenta, a qual pode rodar em uma única máquina *host* ou em várias.

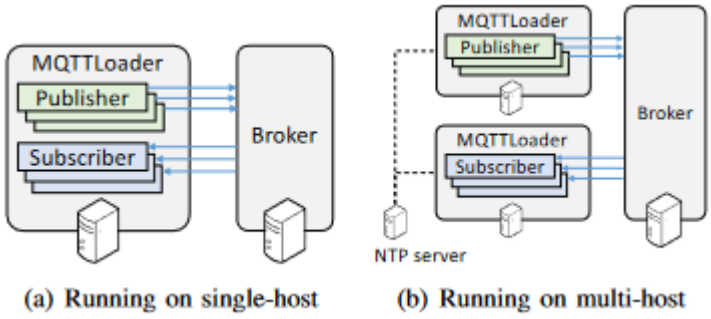

Figura 2 – Visão geral do *MQTTLoader* reproduzido de Banno et al. (2).

Após a execução da ferramenta, resultados de medição sobre o tráfego de mensagens desde o publicador até o assinante serão exibidos. Os resultados fornecidos são informações de estatísticas, como: taxa máxima e média de transferência dos publicadores e assinantes; total de mensagens enviadas e recebidas pelos publicadores e assinantes, respectivamente; e latência máxima e média. Quanto a latência, ela é obtida pela diferença entre o tempo de envio e chegada, já que cada mensagem possui um *timestamp*. No caso em que o *MQTTLoader* roda em diversas máquinas, o cálculo é feito pelo servidor NTP. Sendo assim, as estatísticas facilitam avaliar e monitorar a qualidade e desempenho de determinada aplicação.

5.2 UM MÉTODO DE COLETA DE DADOS DE IOT ESCALÁVEL POR ASSINATURAS COMPARTILHADAS COM *BROKERS* MQTT DISTRIBUÍDOS

Banno; Yoshizawa (1) propõem um método de coleta de dados de IoT escalável com *brokers* MQTT distribuídos. Esse método objetiva solucionar o problema de gargalo que afeta o *broker* ou assinante quando há grande transferência de dados pelos publicadores. A figura 3

mostra uma visão geral do método e utiliza vários *brokers* para distribuir a carga. Além disso, cada aplicativo usa vários assinantes para receber os dados de IoT em paralelo.

Assumindo um grande número de publicadores, o servidor *Domain Name System* (DNS) decide para qual *broker* cada um deles será conectado. Quanto ao número de assinantes, esse valor é maior ou igual ao número de *brokers*. Cada assinante se conecta somente a um *broker* e todos aqueles conectados ao mesmo formam um grupo de assinaturas compartilhadas. Dessa forma, cada mensagem é entregue somente a um membro de cada grupo, balanceando a carga.

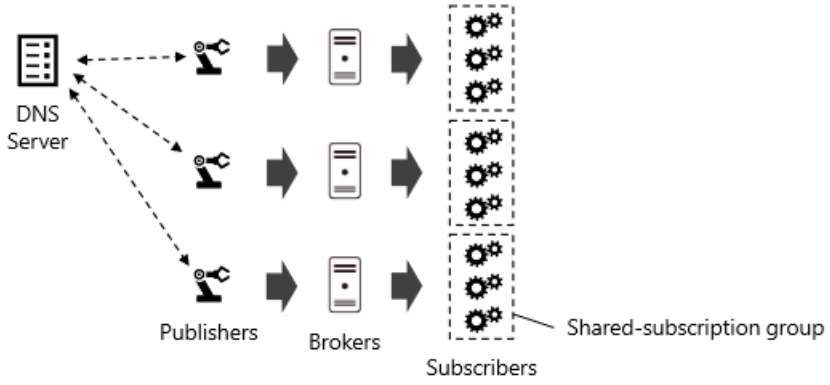

Figura 3 – Visão geral do método proposto reproduzido de Banno; Yoshizawa (1).

O experimento criado utiliza-se da ferramenta mencionada na seção anterior, *MQTTLoader*, para avaliar os *brokers* do MQTT v5.0 e o método proposto. Os parâmetros compreendem a taxa de transferência entre o publicador e o *broker*, a taxa de transferência entre o publicador e o assinante, e o tempo de resposta entre o publicador e o assinante. Os casos de teste comparam as situações: um *broker* B1 e assinante S1; um *broker* B1 e dois assinantes, S1 e S2, que formam um grupo de assinaturas compartilhadas; dois *brokers*, B1 e B2, e dois assinantes, S1 e S2, sem assinaturas compartilhadas; e, o método proposto, dois *brokers*, B1 e B2, e quatro assinantes, S1, S2, S3, S4, dos quais S1 e S2 formam um grupo de assinaturas compartilhadas e S3 e S4 outro.

Quanto aos resultados, o método proposto alcança altas taxas de transferência, sendo a maior nos assinantes. Entretanto, possivelmente devido a esse aspecto, a latência também acaba por ser a maior. Portanto, os resultados do experimento demonstra que o método proposto pode melhorar o rendimento em comparação com as formas convencionais, mas a latência tende a se tornar grande.

5.3 FEDERAÇÃO BASEADA EM MQTT PARA ASSISTÊNCIA HUMANITÁRIA URBANA E OPERAÇÕES DE RECUPERAÇÃO DE DESASTRES

Pradhan (12) apresenta uma arquitetura para federação utilizando diversos clientes e *brokers* que pode ser visualizada pela figura 4. Essa arquitetura objetiva fornecer uma visão para permitir a interoperabilidade federada entre sistemas militares de comando e controle

enquanto faz interface com sistemas civis de tecnologias de informação e comunicação com foco no domínio IoT para futuras operações assistência humanitária e recuperação de desastres (HADR). A implementação se utiliza do MQTT v5.0 e das assinaturas compartilhadas em ambientes HADR e de largura de banda limitada contestados, permitindo reduzir as despesas gerais de tráfego e distribuição.

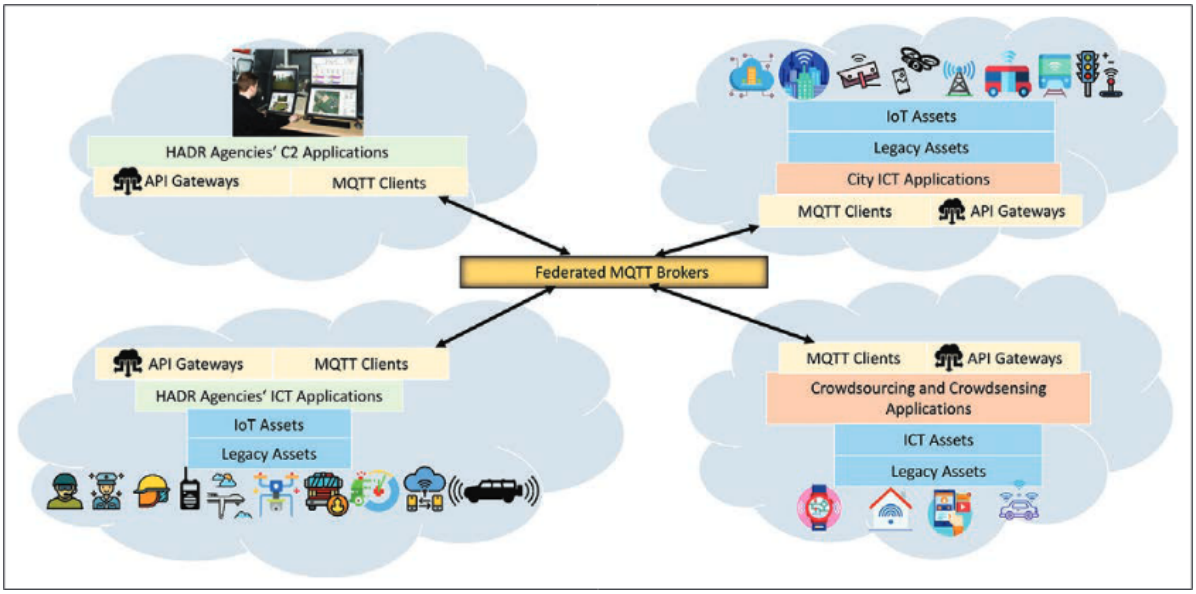

Figura 4 – Representação da arquitetura para troca de consciência situacional baseado em MQTT federado para operações HADR reproduzido de Pradhan (12).

Apesar da arquitetura funcionar bem para o cenário prototípico, em cenários reais a abordagem pode apresentar desvantagens. Foi observado que sempre que um servidor MQTT 5.0 tentava se conectar a um servidor MQTT 3.x utilizando a API do MQTT 5.0, ele retornava um erro informando que a versão do protocolo não é suportada. Isso definitivamente seria um problema, considerando que muitos usuários do MQTT ainda podem estar executando versões mais antigas do *broker*, mesmo que a federação por meio de ontologias padrão possa existir. Além disso, como as conexões entre as entidades utilizam de mecanismos de pontes, esse mecanismo se torna um gargalo para escalabilidade de um sistema federado.

5.4 UMA EXTENSÃO FAIR PARA O PROTOCOLO MQTT

Salami et al. (13) apresentam uma arquitetura de extensão para o protocolo MQTT que valida, normaliza e filtra as mensagens recebidas com base nos princípios de Encontrabilidade (*Findability*), Acessibilidade (*Accessibility*), Interoperabilidade (*Interoperability*) e Reutilização (*Reusability*) (FAIR) para melhorar a interoperabilidade e a reutilização dos dados.

Como os dados recebidos por sistemas IoT são heterogêneos por natureza, cada um com seus padrões, essa arquitetura pode desempenhar um papel crucial na integração desses dados para serem facilmente acessíveis para humanos ou máquinas. Além disso, o modelo também

usa o mecanismo de assinaturas compartilhadas para diminuir a latência e taxa de queda de mensagens. A figura 5 apresenta o modelo proposto.

Para avaliar um sistema com os princípios FAIR foram utilizadas até 22 métricas (figura 6) de avaliação em dois cenários: no primeiro, não há agentes FAIR no sistema; e no segundo, é adicionado um agente FAIR para validar, padronizar e filtrar os dados recebidos. Os resultados mostram que a arquitetura proposta passa com sucesso em 18 indicadores de maturidade de 22, melhora o grau de *FAIRness* (justiça) do sistema e reduz a taxa de mensagens descartadas.

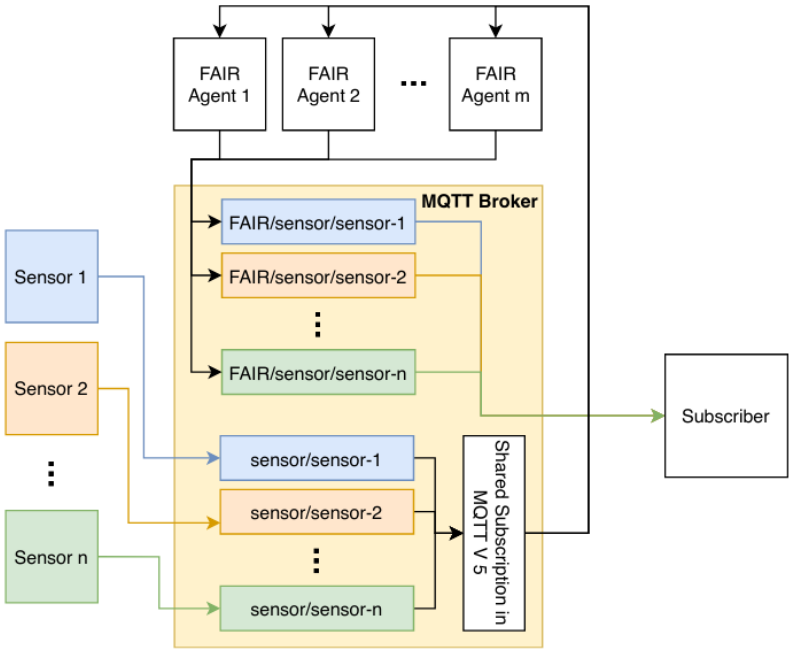

Figura 5 – Representação da arquitetura do sistema proposto reproduzido de Salami et al. (13).

	Metric	Status
F	Identifier Persistence	✗
	Data Identifier Persistence	✗
	Searchable in major search engine	✗
	Unique Identifier	✓
	Structured Metadata	✓
	Grounded Metadata	✓
	Data Identifier Explicitly In Metadata	✓
	Metadata Identifier Explicitly In Metadata	✓
A	Open free protocol for data retrieval	o
	Open free protocol for metadata retrieval	o
	Data authentication and authorization	o
	Metadata authentication and authorization	o
	Metadata Persistence	✗
I	Metadata KRL (weak)	✓
	Metadata KRL (strong)	✓
	Data KRL (weak)	✓
	Data KRL (strong)	✓
	Metadata uses FAIR vocabularies (weak)	✓
	Metadata uses FAIR vocabularies (strong)	✓
	Metadata with qualified outward references	✓
R	Metadata Includes License (weak)	✓
	Metadata Includes License (strong)	✓

Figura 6 – Representação das 22 métricas de avaliação utilizadas para avaliar um sistema com os princípios FAIR reproduzido de Salami et al. (13).

5.5 AGENDANDO MENSAGENS NO GRUPO DE ASSINATURAS COMPARTILHADAS DO MQTT

Matić et al. (7) apresentam o mecanismo *resource-levelling forwarding* (RLF) para enviar mensagens em um grupo de assinaturas compartilhadas do MQTT que se utiliza de uma arquitetura de nuvem em *cluster*. O mecanismo se diferencia de outras soluções para assinaturas compartilhadas, como *round-robin*, *sticky*, *hash* ou *random*, por levar em conta a capacidade de processamento disponível dos serviços do cliente ao encaminhá-lo uma mensagem. A figura 7 mostra a representação do grafo do sistema, sendo os *brokers* as replicações.

O algoritmo tem em vista a capacidade de cada réplica do *broker* e da capacidade dos respectivos clientes conectados, a qual é representada pela quantidade de mensagens processáveis. Logo, o cálculo da capacidade da instância é dado pelo total da capacidade de seus clientes. Assim, a probabilidade para quem a mensagem será encaminhada é dada pela capacidade, primeiramente da réplica e em seguida do cliente.

Quanto aos resultados do teste aplicado ao algoritmo RLF comparado a outros já implementados, percebe-se que ele permite um processamento mais rápido que os demais, sendo assim uma opção viável para as condições do ambiente de testes.

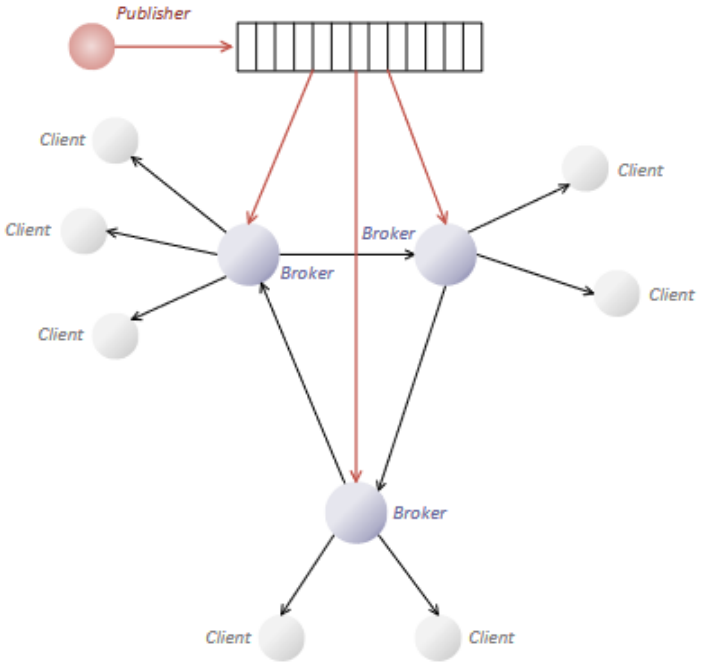

Figura 7 – Representação simplificada do grafo do sistema reproduzido de Matić et al. (7).

5.6 OTIMIZAÇÃO DA COMUNICAÇÃO MQTT ENTRE MICROSSERVIÇOS NA NUVEM IOT

Matic et al. (6) tem por objetivo encontrar um algoritmo para otimizar a comunicação entre microsserviços em nuvem de IoT. Assim sendo, é proposto a utilização de uma arquitetura

de nuvem com grande disponibilidade, onde todas as funcionalidades são implementadas como microsserviços e precisam ser tolerantes a falhas. Considerando a dependência de vários fatores para o objetivo, a solução encontrada foi o desenvolvimento de microsserviços capazes de replicação. Dessa forma, todas as replicações operariam o tempo todo e, caso um serviço ficasse indisponível, outros levariam sua carga.

A partir dessa arquitetura e da finalidade de que apenas uma única replicação processe cada mensagem enviada, o método de assinaturas compartilhadas é utilizado. Entretanto, esse método utiliza de algoritmos do *broker* para o encaminhamento das mensagens, como *random* e *hash*, e nenhum leva em consideração a capacidade de processamento disponível do cliente. A partir disso, um novo algoritmo de agendamento referido como *balanced forwarding* (BF) é proposto. A figura 8 apresenta o grafo do sistema com utilizando assinaturas compartilhadas.

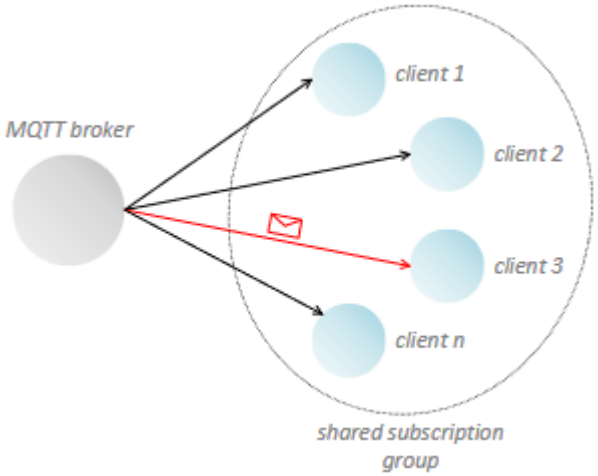

Figura 8 – Representação do grafo do sistema reproduzido de Matic et al. (6).

Dessa maneira, a arquitetura pode ser representada em que cada membro, cliente, do grupo de assinaturas compartilhadas tem uma capacidade, quantidade de mensagens que ele pode processar, e a soma de todas as capacidade do grupo é a capacidade total do *broker* ao qual está conectado. Isto posto, o *broker*, utilizando-se do algoritmo BF, encaminhará a mensagem para o cliente com a maior capacidade disponível no momento. Logo, quando um cliente recebe uma mensagem sua capacidade é reduzida até processá-la e quando finalizada ela é aumentada.

Os testes foram realizados por meio da publicação de mensagens em forma de rajada e o desempenho foi comparado aos algoritmos *random* e *hash*. Quanto aos resultados, o algoritmo conseguiu processar todas as mensagens e sem crescer linearmente, algo não atingido pelos demais. Portanto, o algoritmo BF conseguiu otimizar a comunicação MQTT entre os clientes dentro de um grupo de assinaturas compartilhadas, alcançando seu objetivo.

6 ANÁLISE DE DESEMPENHO

O presente estudo foi do tipo experimental e composto de cenários de testes e ferramentas com diferentes configurações, visando observar o comportamento do MQTT v5.0 em duas situações: com assinaturas compartilhadas e sem assinaturas compartilhadas. A seguir estará descrito a construção do sistema para o experimento, os resultados e a interpretação dos mesmos.

6.1 CENÁRIOS E FERRAMENTAS

6.1.1 *Mosquitto*

O *Mosquitto* (4) é um servidor de mensagens de código aberto que implementa o protocolo MQTT e foi utilizado como o servidor da aplicação para o experimento em sua versão 2.0.11.

Algumas foram configurações necessárias para o pleno funcionamento do servidor. Como o *Mosquitto* permite, assim que instalado, somente conexões locais provindas da própria máquina, foi necessário adicionar a permissão de conexões remotas por meio de um arquivo de configuração customizado. Também foi necessário adicionar a permissão a porta correspondente utilizada pelo servidor nas configurações do *firewall*.

6.1.2 *MQTTLoader*

O *MQTTLoader* é uma ferramenta, desenvolvida na linguagem de programação Java, que suporta o MQTT v5.0 e implementa os clientes MQTT da aplicação e que neste experimento foi utilizada a versão 0.8.4. Ela é responsável pela criação e recebimento das mensagens por parte dos clientes publicadores e assinantes, além de suportar diversas configurações para o que o experimento aconteça de forma correta. Além disso, a ferramenta permite a disponibilização dos resultados em arquivos CSV (*comma-separated values*), os quais foram utilizados para a construção dos resultados do experimento.

6.1.3 Servidor NTP

Um servidor NTP foi configurado para o gerenciamento do cálculo da latência das mensagens pelo *MQTTLoader*. O servidor NTP foi configurado na mesma máquina aonde foram alocados os clientes publicadores das mensagens.

6.1.4 Cenários de testes

Um total de 6 cenários de teste foram criados a fim de testar diferentes configurações do sistema, mas seguindo alguns padrões. Cada cenário foi executado tanto em um ambiente

com e sem assinaturas compartilhadas e cada configuração diferente de teste foi executada por ao menos 30 vezes. Cada cenário é realizado utilizando um sistema com três máquinas, sendo cada uma responsável por uma entidade do sistema: clientes publicadores, servidor e clientes assinantes. Todas as máquinas tem a mesma configuração, segue ela: processador, Intel® Core™ i7-3770; memória RAM, 8 GB DDR3 em *dual-channel*; armazenamento, HD de 500 GB; sistema operacional, Ubuntu 22.04.

Quanto ao montante de mensagens criados por cada publicador, foram definidos quatro números, são eles: 1000, 3750, 14000 e 50000. Cada mensagem tem tamanho fixo de 4096 bytes para o seu conteúdo. Quanto ao nível de QoS, o nível 1 foi utilizado para a publicação e recebimento de mensagens, a fim de evitar a perda de mensagens. Portanto, para cada cenário de teste somente os números dos clientes que variam e são eles que os definem. A tabela abaixo apresenta as configurações de todos os seis cenários de teste:

Cenário de teste	**Publicadores**	**Assinantes**
1	1	2
2	1	3
3	2	2
4	2	3
5	1	1
6	2	1

Tabela 1 – Representação das configurações dos cenários de teste.

6.2 RESULTADOS DOS TESTES

Uma tabela foi construída apresentando um resumo dos resultados obtidos em todos os seis cenários de teste de acordo com a quantidade de mensagens e pode ser visualizada abaixo. Algumas observações: a coluna de título “CT” corresponde ao cenário de teste, enquanto que as colunas de título “Normal” corresponde aos resultados das assinaturas não compartilhadas e de título “Comp.” às assinaturas compartilhadas.

	Tempo de processamento em milissegundos (ms)							
	1000		**3750**		**14000**		**50000**	
CT	**Normal**	**Comp.**	**Normal**	**Comp.**	**Normal**	**Comp.**	**Normal**	**Comp.**
1	666,67	620,53	2127,93	1952,83	6902,20	6305,03	23320,83	21024,23
2	718,00	620,13	2329,47	1956,80	8263,53	6320,50	28629,13	21061,30
3	702,70	578,47	2407,73	1973,07	8405,17	7000,60	28839,13	24256,57
4	823,20	578,77	2791,80	1974,03	9861,23	6849,60	34474,57	23951,73
5	615,17	621,47	1954,83	1938,83	6245,47	6262,00	20819,30	20926,63
6	580,70	574,53	1966,23	1958,97	6874,07	6947,73	24072,13	23979,43

Tabela 2 – Representação dos resultados dos cenários de teste.

6.2.1 Análise dos resultados

A seguir serão apresentados todos os resultados obtidos de acordo com o número de mensagens por meio da representação gráfica dos testes, além da interpretação dos resultados. É interessante destacar de antemão que ao analisar os cenários de teste de 1 a 4, o número de mensagens recebidas para assinaturas sem compartilhamento sempre aumenta ao passar de cenário para outro. Além disso, ao analisar os cenários de teste 5 e 6, ocorre um comportamento comum a todos os gráficos: a diferença no tempo de processamento varia para mensagens com ou sem assinaturas compartilhadas, deixando de seguir uma tendência. Esse comportamento pode ser explicado por dois fatores:

- A forma como o algoritmo de roteamento, utilizado pelo *broker*, trabalha, pois ainda é necessário escolher o destinatário de uma assinatura compartilhada, mesmo que o grupo tenha apenas um assinante. Dessa forma, o algoritmo cria uma latência que pode ser observada em alguns casos nesses dois cenários de teste e explica a diferença no tempo de processamento;
- Os resultados não chegam a serem significativamente distantes entre si, os quais podem serem explicados pela margem de erro do desvio padrão.

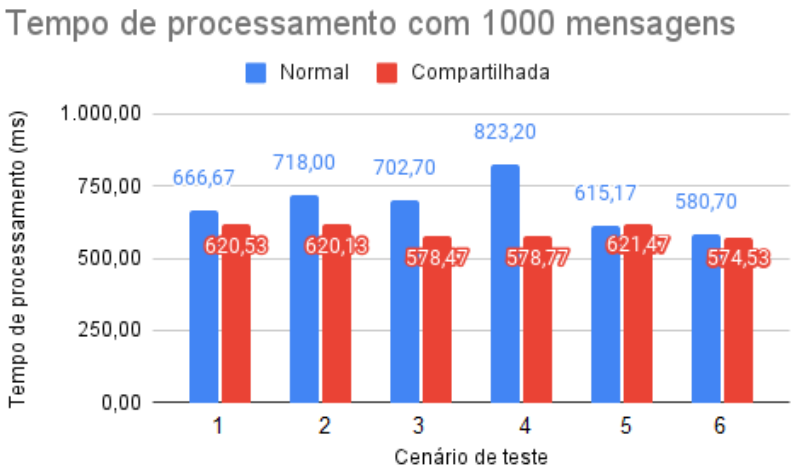

Figura 9 – Representação gráfica dos testes com 1000 mensagens.

De acordo com o gráfico da figura 9 e analisando os cenários de teste de 1 a 4, a diferença dos resultados entre mensagens com e sem assinaturas compartilhadas aumenta ao passar de cenário, visto que o aumento número de mensagens recebidas é mais impactante para assinaturas sem compartilhamento. Já os resultados para as assinaturas compartilhadas entre o cenário 1 e 2 são similares e esse comportamento também ocorre no cenários 3 e 4, devido a mesma quantidade de mensagens recebidas. Quanto aos cenários 5 e 6, os resultados são muito

similares, mas variam e podem ser explicados pelo algoritmo de roteamento e margem de erro do desvio padrão.

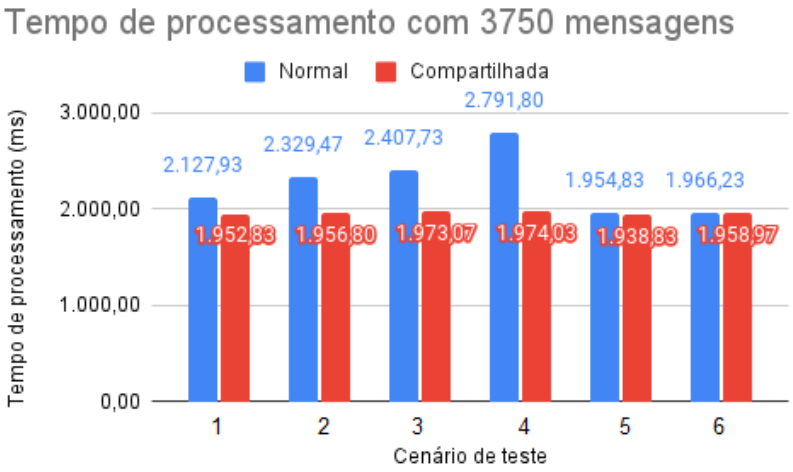

Figura 10 – Representação gráfica dos testes com 3750 mensagens.

De acordo com o gráfico da figura 10 e analisando os cenários de teste de 1 a 4, o tempo de processamento segue uma tendência de aumento para mensagens sem assinaturas compartilhadas, além de ser maior que as assinaturas compartilhadas. Já os resultados para as assinaturas compartilhadas entre o cenário 1 e 2 e 3 e 4 são similares, mesmo comportamento citado anteriormente. Quanto aos cenários 5 e 6, os resultados são melhores para assinaturas compartilhadas, o que pode ser explicado pela margem de erro do desvio padrão.

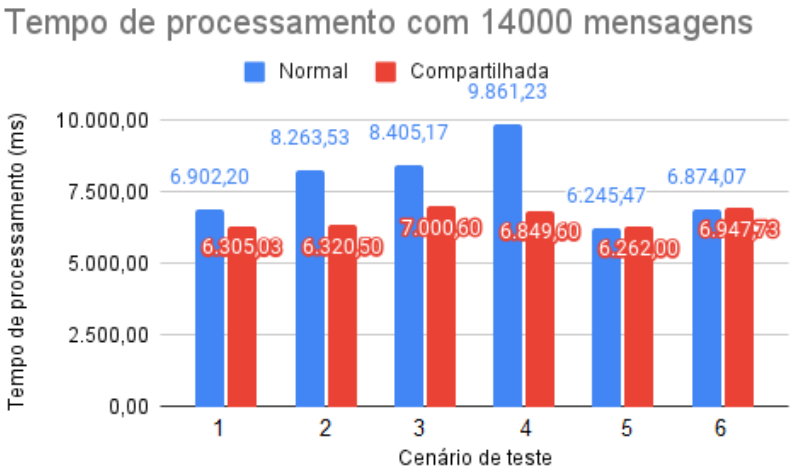

Figura 11 – Representação gráfica dos testes com 14000 mensagens.

De acordo com o gráfico da figura 11 e analisando os cenários de teste de 1 a 4, o tempo de processamento novamente segue uma tendência de aumento para mensagens sem assinaturas compartilhadas. Já os resultados para as assinaturas compartilhadas entre o cenário 1 e 2 são

similares, mas entre o cenário 3 e 4 a diferença é um pouco mais notável, o que pode ser pela margem de erro do desvio padrão. Quanto aos cenários 5 e 6, os resultados são menores para assinaturas sem compartilhamento, o que pode ser explicado pelo algoritmo de roteamento.

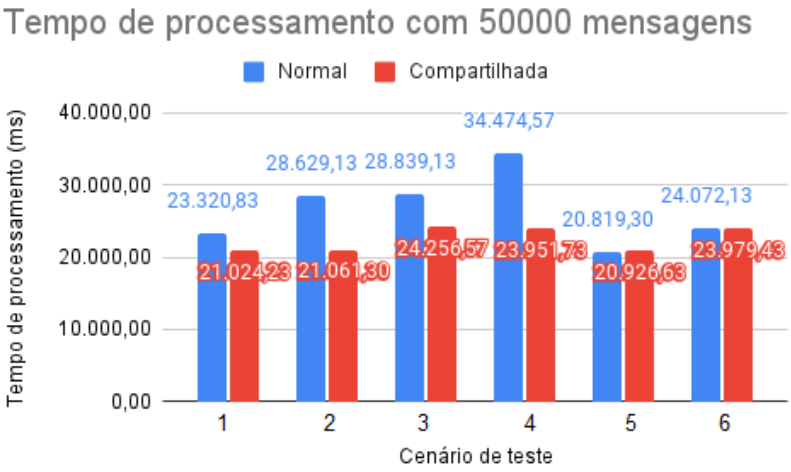

Figura 12 – Representação gráfica dos testes com 50000 mensagens.

De acordo com o gráfico da figura 12 e analisando os cenários de teste de 1 a 4, o tempo de processamento novamente segue uma tendência de aumento para mensagens sem assinaturas compartilhadas. Já os resultados de cenários de teste de 1 a 4 para as assinaturas compartilhadas, o comportamento é similar ao teste com 14000 mensagens da figura 11. Quanto aos cenários 5 e 6, os resultados são muito similares, mas variam e podem ser explicados tanto pelo algoritmo de roteamento como pela margem de erro do desvio padrão.

Portanto, ao analisar todos os resultados, é possível compreender o funcionamento da funcionalidade de assinaturas compartilhadas e que a mesma atinge o seu objetivo de balancear a carga e diminuir o tempo de processamento, já que a quantidade de mensagens recebidas e processadas é reduzida quando a funcionalidade está em uso.

7 CONCLUSÃO

Alcançou-se com o presente trabalho os objetivos propostos tanto de avaliar a funcionalidade de assinaturas compartilhadas no MQTT v5.0 com foco nas características de escalabilidade horizontal, como a aplicação da ferramenta *MQTTLoader*. Conseguiu-se perceber que a funcionalidade de assinaturas compartilhadas realmente faz o que propõe, ou seja, conseguiu diminuir o tempo de processamento devido à diminuição do tráfego de mensagens por meio do balanceamento da carga.

Quanto a avaliação das características de escalabilidade horizontal, os cenários de teste ilustraram diversas configurações possíveis de sistemas. Os resultados dos testes demonstraram que mesmo aumentando o número de assinantes de 2 para 3, por exemplo, a funcionalidade de assinaturas compartilhadas garantiu o balanceamento das mensagens, visto os resultados similares. Além disso, quanto ao *MQTTLoader*, essa ferramento foi muito útil, pois facilitou e permitiu a automatização dos testes e a obtenção dos resultados para a análise do presente trabalho.

7.1 TRABALHOS FUTUROS

Como sugestão para trabalho futuro, a ideia é similar ao do presente trabalho, mas nesse caso o objetivo é avaliar como as assinaturas compartilhadas no MQTT v5.0 afetam o desempenho de um sistema utilizando computadores portáteis (*laptops*) invés de computadores de mesa. Visto que *laptops* tem a característica de serem computadores enxutos, econômicos e que dependem da bateria para o seu funcionamento, o objetivo seria avaliar o quanto um sistema com e sem assinaturas compartilhadas afetaria na duração de funcionamento do sistema levando em conta a duração da bateria.

REFERÊNCIAS

1 BANNO, Ryohei; YOSHIZAWA, Toshinori. A Scalable IoT Data Collection Method by Shared-Subscription with Distributed MQTT Brokers. In: SPRINGER. INTERNATIONAL Conference on Mobile Networks and Management. [S.l.: s.n.], 2021. p. 218–226.

2 BANNO, Ryohei et al. Measuring Performance of MQTT v5. 0 Brokers with MQTTLoader. In: IEEE. 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC). [S.l.: s.n.], 2021. p. 1–2.

3 DISTRIBUTED SYSTEMS GROUP. **MQTTLoader**. [S.l.: s.n.]. Disponível em: <https://github.com/dist-sys/mqttloader>. Acesso em: 24 jun. 2022.

4 ECLIPSE FOUNDATION. **Elipse Mosquitto**. [S.l.: s.n.]. Disponível em: <https://mosquitto.org/>. Acesso em: 2 ago. 2022.

5 _____. **IoT & Edge Developer Survey Report 2021**. [S.l.: s.n.]. Disponível em: <https://iot.eclipse.org/>. Acesso em: 22 mar. 2022.

6 MATIC, Milica et al. Optimization of MQTT Communication Between Microservices in the IoT Cloud. In: IEEE. 2021 IEEE International Conference on Consumer Electronics (ICCE). [S.l.: s.n.], 2021. p. 1–3.

7 MATIĆ, Milica et al. Scheduling messages within MQTT shared subscription group in the clustered cloud architecture. In: IEEE. 2020 28th Telecommunications Forum (TELFOR). [S.l.: s.n.], 2020. p. 1–4.

8 MILEVA, Aleksandra et al. Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. **Computers & security**, Elsevier, v. 104, p. 102207, 2021.

9 MISHRA, Biswajeeban; KERTESZ, Attila. The use of MQTT in M2M and IoT systems: A survey. **IEEE Access**, IEEE, v. 8, p. 201071–201086, 2020.

10 OASIS. **MQTT v3.1.1**. [S.l.: s.n.]. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>. Acesso em: 24 mar. 2022.

11 _____. **MQTT v5.0**. [S.l.: s.n.]. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>. Acesso em: 24 mar. 2022.

12 PRADHAN, Manas. Federation Based on MQTT for Urban Humanitarian Assistance and Disaster Recovery Operations. **IEEE Communications Magazine**, IEEE, v. 59, n. 2, p. 43–49, 2021.

13 SALAMI, Dariush et al. A FAIR Extension for the MQTT Protocol. In: IEEE. 2020 16th International Conference on Mobility, Sensing and Networking (MSN). [S.l.: s.n.], 2020. p. 10–16.