Revista Brasileira de Computação Aplicada, Abril, 2023

DOI: 10.5335/rbca.v15i1.13510
Vol. 15, Nº 1, pp. 22–33
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

# Análise experimental dos protocolos MQTT e MQTT-SN

# Experimental analysis of the MQTT and MQTT-SN protocols

Marco Aurélio Spohn[1], and Willian Bordignon Genero[1]

[1]Universidade Federal da Fronteira Sul

*marco.spohn@uffs.edu.br; willian-b-g@hotmail.com

**Recebido:** 20/05/2022. **Revisado:** 21/02/2023. **Aceito:** 31/03/2023.

## Resumo

A Internet das Coisas (*Internet of Things*, IoT) contempla um número crescente de dispositivos heterogêneos conectados à Internet. Considerando os requisitos inerentes à escalabilidade desses sistemas, torna-se fundamental o emprego de protocolos de comunicação eficientes. O paradigma de comunicação *Publish/Subscribe* (P/S) apresenta baixa complexidade e carga de controle reduzida. No modelo P/S, a fonte (publicador) envia mensagens a um *broker* que, por sua vez, encaminha as mesmas aos clientes interessados (assinantes). Este trabalho apresenta uma análise comparativa entre os protocolos MQTT e sua variante MQTT-SN, ambos baseados no modelo P/S. O primeiro, considera-se um dos protocolos mais empregados no desenvolvimento de aplicações em IoT e, o segundo, trata-se de uma variante do primeiro projetada para o ambiente das redes de sensores. Os resultados produzidos possibilitam identificar os desafios enfrentados por cada um dos protocolos frente um número crescente de clientes e fluxo de mensagens. A qualidade de serviço de melhor esforço tem consequências perceptíveis ao MQTT apenas em momentos de desconexão do cliente, enquanto no MQTT-SN é um fator de atenção contínuo em decorrência do emprego do UDP. Apesar do MQTT apresentar, consideravelmente, menos perdas de pacotes em cenários de maior demanda agregada, observa-se um crescimento acentuado na latência. Quanto menor a tolerância à perda de pacotes, torna-se menos recomendado o emprego do MQTT-SN.

**Palavras-Chave**: Modelo de Comunicação Publicador/Assinante; MQTT; MQTT-SN.

## Abstract

The Internet of Things (IoT) comprises many heterogeneous devices connected to the Internet. Considering the requirements inherent to these systems' scalability, efficient communication protocols are essential. The Publish/Subscribe (P/S) communication paradigm has low complexity and reduced control overhead. In the P/S model, the source (publisher) sends messages to a broker, which, in turn, forwards them to interested customers (subscribers). This work presents a comparative analysis between the MQTT protocol and its MQTT-SN variant, both based on the P/S model. The first is considered one of the most used protocols in developing IoT applications, and the second is a variant of the first designed for sensor networks. The results allow us to identify the challenges faced by each of the protocols in the face of a growing number of clients and message flow. The best-effort quality of service has noticeable consequences for MQTT only in moments of client disconnection, while MQTT-SN is intrinsically best-effort due to the use of UDP. Although MQTT presents considerably fewer packet losses in higher aggregate demand scenarios, a sharp latency increase is observed. The lower the packet loss tolerance, the less recommended MQTT-SN is.

**Keywords**: MQTT; MQTT-SN; Publish/Subscribe Communication Model.

## 1 Introdução

A Internet das Coisas (*Internet-of-Things, IoT*) pode ser definida como um conjunto de paradigmas clássicos pre-

sentes na Internet portados para ambientes e dispositivos outrora não conectados à rede (Al-Fuqaha et al., 2015). Independente da aplicação final, os dispositivos e demais sistemas envolvidos nesse ecossistema necessitam se comunicar. Nesse contexto, há várias alternativas de protocolos a nível de aplicação que se destacam no desenvolvimento de soluções em IoT. Muitos desses protocolos compartilham o paradigma de comunicação *Publish/Subscribe* (P/S).

No modelo P/S, os clientes não se comunicam diretamente, mas sim por intermédio de um *broker*. As mensagens que são trocadas pelos clientes são comumente rotuladas em tópicos. Os clientes que são originadores das mensagens são denominados *publishers* (publicadores). Os clientes interessados nas mensagens são denominados *subscribers* (assinantes). Em síntese, os publicadores enviam as mensagens ao *broker* e este as repassa aos assinantes correspondentes.

O protocolo MQTT (Standard, 2020) é um dos mais adotados no desenvolvimento de aplicações em IoT. O suporte à implementação dos clientes é amplo em termos de sistemas operacionais e linguagens de programação. A facilidade de projeto e implementação dos clientes bem como a disponibilidade de uma variedade de *brokers* de código livre são os principais fatores que impulsionaram sua adoção.

O MQTT atende, em geral, os requisitos limitados de recursos da maioria dos dispositivos empregados em IoT. Por suportar três níveis de qualidade de serviço, permite tratar desde aplicações que sobrevivem com a comunicação de melhor esforço até aquelas que exigem confiabilidade na entrega das mensagens. Outro aspecto essencial do protocolo é o suporte a comunicação assíncrona, muitas vezes um requisito essencial pois não se pode garantir que os dispositivos estejam permanentemente ligados.

Para ambientes de redes de sensores, onde os dispositivos são geralmente ainda mais limitados em recursos, há o protocolo MQTT-SN. Este possibilita a comunicação dos dispositivos finais empregando o protocolo de transporte UDP. Como o MQTT é originalmente baseado no TCP, há uma entidade intermediária, denominada *gateway*, fazendo a ponte na comunicação entre os dispositivos finais e o *broker* MQTT. Além do emprego de um protocolo de transporte menos custoso, mas do tipo melhor esforço, há suporte a mensagens de tamanho reduzido, encaixando-se nas restrições típicas de ambientes com canais de comunicação de largura de banda estreita.

Há um número reduzido de trabalhos científicos comparando os protocolos MQTT e MQTT-SN. A maioria desses trabalhos não realizam uma avaliação extensa e, geralmente, dispersam sobre um conjunto maior de protocolos sem um maior aprofundamento. Neste trabalho, foca-se apenas nos dois protocolos, ampliando-se o conjunto de cenários de teste. Alguns dos resultados obtidos corroboram aqueles poucos apresentados em trabalhos relacionados. Em síntese, os resultados permitem identificar o padrão de comportamento de ambos os protocolos para uma demanda crescente de clientes e fluxo de mensagens.

O restante desse artigo contempla, inicialmente, uma breve visão sobre os principais protocolos de camada de aplicação empregados no desenvolvimento de soluções em IoT. Em seguida, elencam-se os principais trabalhos relacionados. Após detalhar a metodologia empregada nesse trabalho, apresentam-se os resultados e a análise de desempenho dos dois protocolos. Por final, descrevemos as nossas conclusões.

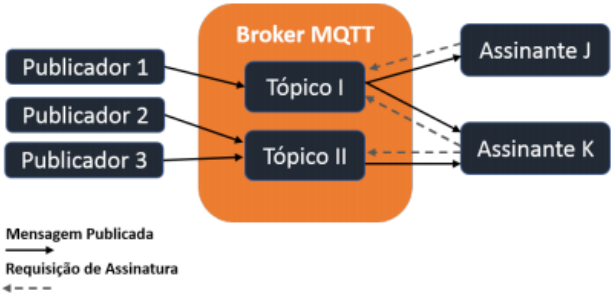


**Figura 1:** Funcionamento do protocolo MQTT (Fonte: Quincozes (Quincozes et al., 2019)).

## 2 Protocolos de aplicação em IoT

Além dos protocolos alvo desse trabalho, apresenta-se breve descrição dos principais protocolos, a nível de aplicação, mais empregados no desenvolvimento de soluções em IoT.

### 2.1 MQTT

Desenvolvido por Andy Stanford-Clark e Arlen Nipper em 1999, o MQTT (Standard, 2020) é um protocolo para camada de aplicação amplamente empregado na comunicação *Machine-to-Machine* (M2M), *Server-to-Server* (S2S) e *Machine-to-Server* (M2S) na Internet das Coisas. A utilização em aplicações por grandes empresas (e.g., Facebook) e o suporte às plataformas de *cloud computing* Amazon AWS, Google Cloud, IBM Cloud e Microsoft Azure alavancaram a expansão do protocolo, tornando-o um dos mais utilizados no desenvolvimento de aplicações em IoT.

O MQTT utiliza o paradigma de comunicação *Publish/-Subscribe* (Karagiannis et al., 2015). Neste modelo, os clientes não se comunicam diretamente, mas sim por intermédio de um *broker*/servidor. Um cliente que é fonte de informação é denominado publicador. Este, por intermédio de um *broker*, comunica-se com as entidades interessadas na informação, denominadas assinantes. As mensagens são rotuladas como tópicos em uma estrutura hierárquica, facilitando o gerenciamento e encaminhamento de mensagens.

O MQTT contempla três entidades elementares (Fig. 1): *subscriber*, *publisher* e o *broker*. Os clientes assinantes (*subscribers*) inscrevem-se em tópicos de mensagens no *broker*. As mensagens enviadas pelos publicadores (*publishers*) para o *broker* são, posteriormente, encaminhadas para todos os assinantes correspondentes. Esse formato de comunicação pode suprir os mecanismos de roteamento *one-to-one*, *one-to-many* e *many-to-many* (Yassein et al., 2017).

Uma mensagem MQTT (Fig. 2) possui um cabeçalho fixo de dois bytes contendo o tipo de pacote (conexão, assinatura, publicação, entre outros 12 tipos), *flags* (para verificar duplicações, qualidade de serviço e retenção no

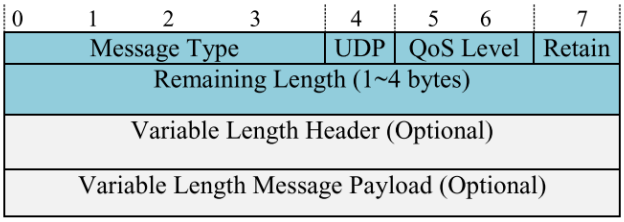


**Figura 2:** Formato de uma mensagem MQTT (Fonte: Al-Fuqaha (Al-Fuqaha et al., 2015)).

*broker*) e o tamanho restante da mensagem. Um cabeçalho opcional pode ser adicionado, assim como a carga útil da mensagem (Banks et al., 2019).

O MQTT oferece três níveis de qualidade de serviço:

- Zero - No máximo uma vez: segue o modelo de melhor esforço (*best effort*), onde a mensagem é entregue uma única vez ou sequer chega ao destino;
- Um - Pelo menos uma vez: a mensagem é reenviada até que se receba a confirmação de recepção, mesmo que ocasione redundância;
- Dois - Exatamente uma vez: sempre entrega uma única vez, tornando-se o modo mais confiável apesar de mais lento.

Por possibilitar comunicação assíncrona entre os clientes e, sobretudo, suporte a mensagens curtas, o MQTT torna-se adequado para dispositivos de recursos limitados, com enlaces com pequena largura de banda, alta latência e/ou não confiáveis, com restrição energética e requisitos de escalabilidade (Quincozes et al., 2019) (Karagiannis et al., 2015).

## 2.2 MQTT-SN

Extensão do MQTT para redes de sensores, o MQTT-SN (*Sensor Networks*) foi projetado para dispositivos extremamente limitados em processamento, armazenamento, e em capacidade energética. Por serem utilizados em locais remotos, ou de difícil acesso e/ou manuseio, os dispositivos de sensores sem fio apresentam restrições quando submetidos a uma demanda de comunicação crescente em volume de dados, dado que isto pode comprometer a vida útil dos dispositivos por exaustão da capacidade das baterias. Neste contexto, o paradigma P/S favorece a comunicação dos dispositivos pela sua baixa complexidade em carga de controle. Adicionalmente, o suporte a comunicação assíncrona é um diferencial do MQTT/MQTT-SN considerando que os dispositivos sensores permanecem a maior parte do tempo em modo de repouso (i.e., *sleep*). Por ser uma extensão do MQTT, o novo protocolo abrange também a integração de ambos (Hunkeler et al., 2008).

O tamanho de uma mensagem MQTT-SN foi reduzido para dar suporte a protocolos de camadas inferiores que oferecem largura de banda limitada como, por exemplo, o padrão IEEE 802.15.4 que provê uma largura de banda máxima de 250 kbit/s (Silva et al., 2019).

Tendo-se como referência a ilustração apresentada na Fig. 3, além das entidades já presentes no MQTT, o MQTT-SN emprega um *gateway* que fica responsável pela interoperabilidade entre os dispositivos internos e externos à rede de sensores (Quincozes et al., 2019). O *gateway* pode ser configurado segundo duas modalidades: "transparente", onde para cada cliente MQTT-SN é mantido uma conexão com o *broker*; e "agregado", onde uma única conexão é compartilhada entre todos os clientes e o *broker* que, apesar de tornar mais complexa a implementação, torna-se mais escalável.

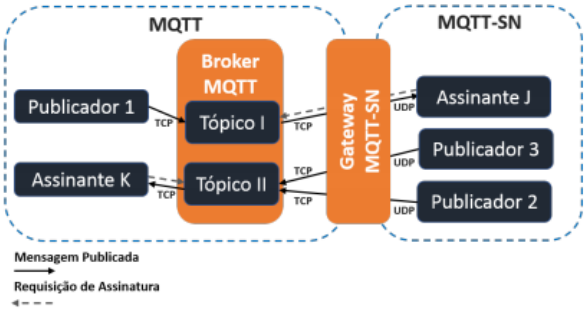


**Figura 3:** Funcionamento do protocolo MQTT-SN (Fonte: Quincozes (Quincozes et al., 2019)).

Outro diferencial do MQTT-SN é o emprego do UDP para camada de transporte: por ser um protocolo não orientado a conexão, possibilita que os dispositivos gerenciem suas interfaces de rede de forma livre e sem implicações diretas na camada de aplicação (i.e., pode-se ligar e desligar a interface de forma mais dinâmica, economizando energia) (Schütz et al., 2017). Além dos níveis de qualidade de serviço disponíveis no MQTT, tem-se o nível -1, que dá suporte ao serviço não orientado a conexão (Viriato et al., n.d.).

Quando o tamanho reduzido do pacote se torna problemático, algumas soluções (Hunkeler et al., 2008) adotam a estratégia de renomear os tópicos com um identificador de dois bytes, permitindo a comunicação mesmo em ambientes extremamente limitados em largura de banda .

## 2.3 CoAP

O *Constrained Application Protocol* (CoAP) foi criado pelo *Internet Engineering Task Force* (IETF), contemplando a transferência de dados baseada no *Representational State Transfer* (REST) para facilitar a interoperabilidade com o HTTP e possibilitar o consumo de serviços utilizando *Uniform Resource Identifiers* (URIs) (Santos et al., 2016). Semelhante ao MQTT-SN, também é baseado no protocolo de transporte UDP. Além do roteamento padrão *unicast*, o protocolo oferece suporte ao roteamento de mensagens via *multicast*.

Como o protocolo é baseado no UDP, a funcionalidade de entrega confiável é implementada a nível de aplicação empregando o protocolo ARQ *Stop-and-Wait*. Apesar de garantir a corretude de confiabilidade, essa estratégia de retransmissão é extremamente ineficiente.

Em termos de segurança, o CoAP sugere utilizar o *Datagram Transport Layer Securety* (DTLS) para proteger as transações. No entanto, o DTLS não foi projetado para IoT,

resultando em obstáculos como a falta de suporte ao roteamento *multicast* e a necessidade de *handshake*, acarretando um custo operacional adicional responsável por um maior consumo de energia e de largura de banda (Karagiannis et al., 2015),.

### 2.4 DDS

O *Data-Distribution Service* (DDS) provê comunicação *machine-to-machine* baseada no paradigma P/S, projetado para facilitar a comunicação distribuída eficiente com ênfase em sistemas de tempo real (Pardo-Castellote, 2003). O DDS se destaca em relação aos demais protocolos ao prover 23 níveis de QoS, contemplando critérios de segurança, urgência, prioridade, durabilidade, confiabilidade, entre outros (Al-Fuqaha et al., 2015).

No DDS, o descobrimento da topologia de rede é dinâmico, sem a necessidade de um *broker* como no MQTT e, desta forma, eliminando a possibilidade de ponto único de falha (Aures and Lübben, 2019). Cinco componentes são necessários para um fluxo completo de dados, empregando o formato de entrega centrado em dados:

- *Publisher*: responsável pela distribuição de dados;
- *DataWriter*: atua entre a aplicação e o *publisher*, analisando aspectos de conformidade à estrutura do tópico;
- *Subscriber*: disponibiliza os dados para as aplicações;
- *DataReader*: recebe os dados provenientes da rede;
- *Topic*: identificador de conteúdo para publicação e assinatura.

### 2.5 AMQP

O *Advanced Message Queuing Protocol* (AMQP) é um padrão aberto de comunicação orientado a mensagens, originalmente proposto para a indústria de sistemas financeiros, em 2006 (Al-Fuqaha et al., 2015). Suporta dois tipos de mensagens: "cruas", as mais simples e com um tamanho reduzido; e, as "anotadas", que permitem acrescentar um cabeçalho e informações sobre durabilidade, tempo de vida, prioridade, entre outras.

O funcionamento é semelhante ao MQTT, com destaque a duas estruturas internas ao *broker*: uma fila de mensagens para armazenar e entregar em ordem sequencial; e, uma entidade denominada *exchange*, que é responsável por fazer o encaminhamento das mensagens para suas devidas filas (Subramoni et al., 2008). O protocolo também oferece confiabilidade na entrega de mensagens com três níveis de qualidade, sendo elas: ao menos uma, no mínimo uma ou exatamente uma (Al-Fuqaha et al., 2015).

### 2.6 XMPP

Desenvolvido pela comunidade Jabber para oferecer um protocolo aberto, seguro, descentralizado e livre de *spam*, o *Extensible Messaging and Presence Protocol* (XMPP) tornou-se padrão do grupo IETF e é utilizado para *chat*, voz, vídeo e telepresença entre múltiplas aplicações (Al-Fuqaha et al., 2015).

Mecanismos de segurança são oferecidos via criptografia SSL/TLS e autenticação com *Simple Authentication and Security Layer* (SASL). Apesar do uso do formato *Extensible Markup Language* (XML) fornecer uma boa interoperabilidade, o protocolo peca em aspectos essenciais para IoT como, por exemplo, o alto consumo de CPU e largura de banda, e nenhum suporte a QoS (Quincozes et al., 2019).

## 3 Trabalhos relacionados

Talaminos-Barroso et al. (2016) empregaram 41 máquinas para realizar um extenso *benchmark* com objetivo de avaliar o comportamento dos protocolos de aplicação baseados no paradigma de comunicação P/S em uma aplicação de *eHealth* para reabilitação de respiração. Tal reabilitação pode acontecer tanto no hospital quanto em casa, e consiste em atividades físicas aeróbicas com monitoramento de batimentos cardíacos, saturação de oxigênio do sangue e consumo de calorias, necessitando-se acompanhamento médico.

Para os protocolos DDS, MQTT, AMQP, XMPP, CoAP e *Java Message Service* (JMS), cinco métricas foram avaliadas: uso de CPU, memória, consumo de largura de banda, latência e *jitter*. As mensagens enviadas não tem retorno de confirmação.

O fato do DDS ser distribuído, resultou em um maior consumo de recursos conforme aumento de mensagens comparado aos demais. Por outro lado, ele apresentou o melhor resultado em latência e *jitter*. O XMPP apresentou desempenho inferior aos demais em todas as métricas. AMQP, CoAP e MQTT tiveram desempenho semelhante entre si, sobretudo em decorrência da utilização de menos recursos.

Mun et al. (2016) testaram o tempo de transmissão, eficiência energética e uso de CPU para os protocolos CoAP, MQTT e MQTT-SN. As mensagens, que podem ser de diferentes tamanhos, partindo de 128 até 1920 bytes, são enviadas para três servidores (Local Utah-EUA, Amazon Web Service Oregon-EUA e Amazon Web Service Tóquio-Japão) que respondem com uma confirmação.

O tempo de transmissão é semelhante entre CoAP e MQTT. O MQTT-SN, por sua vez, apresenta o pior desempenho em todas as métricas. Os autores atribuem o desempenho deficiente à implementação precária da biblioteca Eclipse Paho MQTTSN. Tanto em termos de consumo de energia quanto de CPU, o CoAP é superior ao MQTT para pacotes menores que 1024 bytes, e inferior para pacotes maiores, pois o CoAP fragmenta mensagens maiores de 1kB.

Naik (2017) publicou uma avaliação técnica envolvendo os protocolos AMQP, CoAP, MQTT e HTTP. O HTTP, por não ser projetado para as restrições da IoT, apresenta o maior tamanho de mensagem, consumo de processamento, largura de banda, latência e menor confiabilidade; por outro lado, destaca-se em interoperabilidade, recursos adicionais e padronização.

No aspecto tamanho de mensagem e sobrecarga, o CoAP, apesar de ter um cabeçalho maior que o MQTT, possui uma mensagem menor por operar sobre UDP. Por sua vez, o AMQP apresenta mensagens mais longas devido ao suporte a segurança, confiabilidade, entre outros. Fato este que resulta em um maior consumo de processamento e largura de banda. O CoAP e o MQTT apresentam consumo de recursos semelhantes. O emprego do TCP acarreta em

maior latência de transmissão para o MQTT, AMQP e HTTP. Os três níveis de qualidade de serviço garantem ao MQTT maior confiabilidade entre os quatro protocolos, seguido pelo AMQP. Em contrapartida, o AMQP oferece mais funcionalidades de segurança.

Motivados em melhorar a comunicação de aplicações robóticas que geralmente empregam o protocolo HTTP, Amaran et al. (2015) compararam os protocolos CoAP e MQTT-SN incentivados pelo fato de ambos serem baseados no UDP. Para analisar o tempo de transmissão, 10.000 mensagens foram enviadas entre dois dispositivos configurados em rede local. Constatou-se que a primeira mensagem transmitida foi 59% mais rápida com o CoAP. Esse comportamento é explicado devido ao MQTT-SN necessitar registrar um tópico antes de compartilhar dados. Já para o tempo médio de transmissão, o MQTT-SN executou 30% mais rápido que o CoAP.

Aures and Lübben (2019) avaliam tecnicamente o DDS, MQTT e VSL (Virtual State Layer) do ponto de vista de desenvolvedores e apontam os pontos fortes e fracos de cada um deles. Apesar de ser superado pelo VSL, o DDS apresenta robustez em integridade de dados, autenticação, controle de acesso e na estruturação com tratamento agnóstico de dados. Ambos oferecem criptografia e serialização. A qualidade de serviço, monitoramento e gerenciamento em tempo real é melhor por parte do DDS. Em contraponto, a sobrecarga dos dois é um ponto negativo, assim como a complexidade de uso do DDS. O MQTT, por sua vez, oferece simplicidade e baixa sobrecarga de controle, mas ao custo de funcionalidades limitadas comparadas aos recursos dos demais protocolos.

Usando sensores para coletar dados de frequência cardíaca, oxigênio no sangue, temperatura, resistência e condutividade elétrica da pele, orientação e acelerômetro, Chen and Kunz (2016) observaram a execução do CoAP, DDS, MQTT e uma versão autoral denominada *Custom UDP* em um cenário de aplicação *eHealth*. O ambiente é configurado usando um *laptop*, Arduino Uno e Raspberry Pi 2 conectados em uma rede simulada, limitada e com baixa confiabilidade, sendo que para cada teste são transmitidos 981.600 bytes durante um intervalo de 10 minutos.

O consumo de largura de banda para o CoAP e *Custom UDP* se mantém o mesmo independente da quantidade de pacotes perdidos ou da latência da rede, justamente pelo emprego do protocolo de transporte UDP. Em contrapartida, o DDS e o MQTT (ambos baseados no TCP) apresentam um consumo de largura de banda diferenciado. A latência experimentada pelo MQTT tem uma taxa de crescimento acentuada à medida que a latência do sistema ou perda de pacotes aumentam. O CoAP, DDS e *Custom UDP* apresentam um desempenho semelhante na latência.

Os protocolos baseados em TCP realizam retransmissões para pacotes perdidos na rede. Tal comportamento não se repete para os protocolos baseados em UDP. Para a métrica de consumo de rede, a cada 100 bytes de *payload* outros 289 bytes são usados para controle no DDS, 76 no MQTT, 36 no CoAP e 32 no *Custom UDP*. Em compensação, o DDS apresenta aproximadamente metade da latência média do MQTT; porém, o CoAP e *Custom UDP* continuam melhores nesse quesito.

Com o objetivo de auxiliar desenvolvedores na escolha de protocolos a nível de aplicação, Cosmi and Mota (2019) mensuraram qualitativa e quantitativamente os protocolos AMQP, CoAP, MQTT e MQTT-SN. Na parte técnica foram examinados aspectos como restrições de processamento, qualidade de sinal de rede, suporte a grande volume de dados, entre outros. Destacando-se dos demais, o AMQP exige mais recursos para oferecer melhores funcionalidades. O CoAP, MQTT e MQTT-SN possuem foco em dispositivos de recursos limitados, mas divergem em características como, por exemplo, suporte a envio de dados em tempo real, reconhecimento de outros dispositivos e confiabilidade de entrega.

Empregando um ESP8266 como dispositivo de IoT, observou-se melhores resultados do MQTT no quesito de retransmissões para troca de mensagens em um intervalo de tempo menor que 500 ms. O CoAP mostrou-se sensível para intervalos menores que 500 ms. De acordo com o crescimento do *payload* da mensagem, o MQTT diminui a taxa de retransmissões, enquanto o CoAP mantém uma média mais elevada dessa métrica. Os autores também avaliaram o *Round Trip Time* em uma rede sem e com tráfego de fundo: tanto o CoAP como o MQTT mostraram um aumento considerável no RTT ao aumentar a frequência de mensagens.

Chaudhary et al. (2017) em seu estudo buscaram entender o impacto da transmissão de dados entre dispositivos IoT e servidores externos utilizando redes cabeadas, WiFi, 2G, 3G e 4G para os protocolos CoAP, MQTT e AMQP. Foi observado que o número de pacotes gerados para entregar uma mensagem aumenta significativamente quando é requisitado ao MQTT nível QoS 1 ou 2, o mesmo se repetindo para a rede 3G com perdas de dados. A vazão de poucas mensagens é similar entre os protocolos, mas para grande volumes de dados a qualidade de serviço do MQTT suporta maior vazão devido, sobretudo, às possíveis perdas de pacotes e as consequentes retransmissões. Tais fatos replicam-se no consumo de largura de banda, pois para retransmitir um maior volume de mensagens um número crescente de pacotes são transmitidos pela rede.

## 4 Metodologia

O presente trabalho busca de forma experimental analisar aspectos de desempenho dos protocolos MQTT e MQTT-SN no contexto da Internet das Coisas. A análise de dados, do tipo quantitativa, é realizada considerando-se as métricas de latência e perda de pacotes.

### 4.1 Instrumentos e ferramentas de desenvolvimento

#### *4.1.1 Brokers*

**Mosquitto** (Foundation, 2021) é uma implementação de código aberto de servidor MQTT que suporta as versões 3.1, 3.1.1 e 5.0. O projeto integra a *Eclipse Foundation* e está disponível para uma ampla gama de dispositivos e sistemas operacionais. Neste trabalho, emprega-se a versão 1.6.9 do Mosquitto.

Dado que o **Mosquitto** não suporta o protocolo MQTT-SN, o **RSMB** (Eclipse, 2013) surge como solução apropriada pois é uma extensão da implementação do Mosquitto, estando igualmente integrado à comunidade *Eclipse*. O

mesmo projeto fornece o *gateway* necessário para converter mensagens MQTT-SN em MQTT. Neste trabalho, adota-se a versão do RSMB com *git commit id 36fd4ba*.

#### 4.1.2 *Clientes*

A comunidade MQTT oferece várias plataformas para utilizar como clientes. No presente trabalho, adota-se a biblioteca para clientes do projeto Eclipse Paho (Eclipse, 2016). A biblioteca fornece suporte completo para a instanciação de assinantes e publicadores.

Por outro lado, há poucas alternativas para desenvolvimento de clientes do MQTT-SN. Nesse trabalho, a implementação dos clientes é baseada na solução de Humfrey (2013). Algumas alterações foram necessárias no projeto inicial, principalmente referentes ao *payload* das mensagens.

#### 4.1.3 *Hospedagem*

Tanto o servidor **Mosquitto** quanto o **RSMB** são hospedados na nuvem, mais especificamente em Virgínia do Norte - EUA. O serviço usado é o Amazon Web Service EC2, contemplando uma máquina virtual com um núcleo Intel Xeon, um GiB de memória RAM, executando o sistema operacional Ubuntu 20.4. Adotamos o uso da AWS por oferecer uma modalidade de acesso gratuito, facilitando que terceiros possam replicar a avaliação usando a mesma configuração.

Os clientes, publicadores e assinantes, são executados em máquina fisicamente localizada em Santa Catarina - Brasil. Esta máquina contém oito núcleos Apple M1, oito GB de memória RAM, executando o sistema operacional macOS Monterey 12.1.

#### 4.1.4 *Configurações de rede*

As características do enlace de saída da máquina local consistem em comunicação Wi-Fi 5Ghz (i.e., IEEE 802.11a) entre a máquina e o roteador, e via fibra ótica até o provedor. A conexão contratada é de 200 Mb/s para *download* e 100 Mb/s para *upload*. A AWS não fornece informações detalhadas sobre seus enlaces de comunicação, apenas destaca que é de baixo desempenho para o serviço ofertado gratuitamente. No entanto, o *Round Trip Time* (RTT) médio com a VM na Amazon foi avaliado em 147,68 ms (previsível, considerando-se a distância geográfica entre as partes comunicantes).

### 4.2 Cenários de avaliação

Os cenários consistem em configurações de testes que permitem avaliar o desempenho dos protocolos MQTT e MQTT-SN nas métricas de latência e perdas de pacotes. Os parâmetros ajustados são a quantidade de mensagens enviadas e o número de publicadores e assinantes.

Na Tabela 1 estão descritos todos os parâmetros empregados em cada cenário, iniciando-se com uma configuração que demanda poucos recursos e finalizando com uma que estressa o broker. Os cenários foram definidos usando as cardinalidades 1:N e N:N. O número máximo de

[1]Todos os clientes hospedados em máquina AWS.

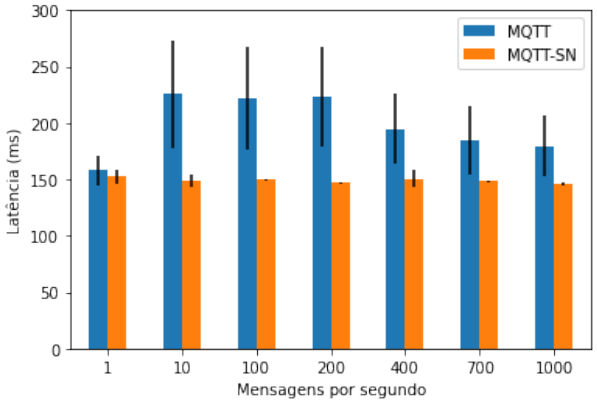


**Figura 4:** Latência: cenário 1 (1 P : 1 A)

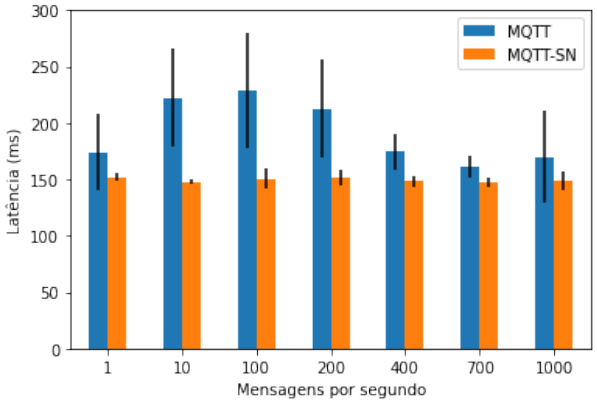


**Figura 5:** Latência: cenário 2 (2 P : 1 A)

16 clientes foi estabelecido após indicativos de um possível gargalo no ambiente de testes.

Todas as mensagens tem 20 bytes, sendo enviadas durante um período de 10 segundos para um mesmo tópico e configuradas com QoS de nível 0. Mensagens que não alcançam um assinante em um período de 30 segundos serão consideradas perdidas. Para cada configuração, calcula-se a média e respectivo desvio padrão das métricas referente a 10 execuções.

## 5 Resultados e análise

Inicialmente, destaca-se que se apresentam os resultados gráficos referentes às perdas de pacotes para os cenários onde estas ocorrem de forma significativa. Já se antecipa que situações de gargalo impactam mais diretamente o protocolo MQTT-SN, considerando que este executa sobre o protocolo UDP.

Ponderando o primeiro cenário como o que demanda a menor carga, os resultados (Fig. 4) são estáveis, com uma latência média do MQTT-SN de aproximadamente 150 ms e de 200 ms para o MQTT. Os resultados do segundo (Fig. 5) e terceiro (Fig. 6) cenário mantiveram-se semelhantes ao primeiro.

No quarto cenário (Fig. 7), identifica-se uma primeira

**Tabela 1:** Tabela com os parâmetros dos cenários avaliados

| Cenário | N° assinantes | N° publicadores | N° msg/s |
|---|---|---|---|
| 1 | 1 | 1 | (1, 10, 100, 200, 400, 700, 1000) |
| 2 | 1 | 2 | (1, 10, 100, 200, 400, 700, 1000) |
| 3 | 1 | 4 | (1, 10, 100, 200, 400, 700, 1000) |
| 4 | 1 | 8 | (1, 10, 100, 200, 400, 700, 1000) |
| 5 | 1 | 16 | (1, 10, 100, 200, 400, 700, 1000) |
| 6 | 2 | 1 | (1, 10, 100, 200, 400, 700, 1000) |
| 7 | 4 | 1 | (1, 10, 100, 200, 400, 700, 1000) |
| 8 | 8 | 1 | (1, 10, 100, 200, 400, 700, 1000) |
| 9 | 16 | 1 | (1, 10, 100, 200, 400, 700, 1000) |
| 10 | 2 | 2 | (1, 10, 100, 200, 400, 700, 1000) |
| 11 | 4 | 4 | (1, 10, 100, 200, 400, 700, 1000) |
| 12 | 8 | 8 | (1, 10, 100, 200, 400, 700, 1000) |
| 13 | 16 | 16 | (1, 10, 100, 200, 400, 700, 1000) |
| 14 [1] | 16 | 16 | (1, 10, 100, 200, 400, 700, 1000) |

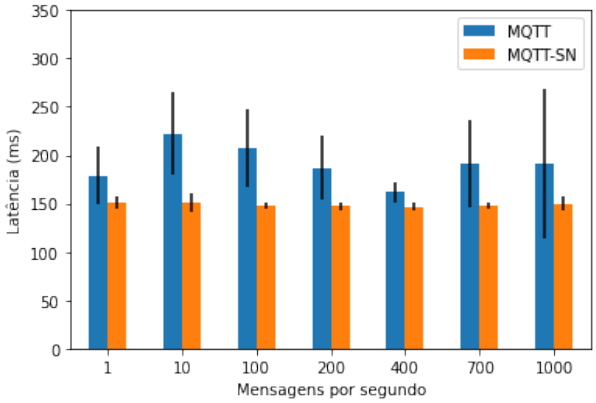


**Figura 6:** Latência: cenário 3 (4 P : 1 A)

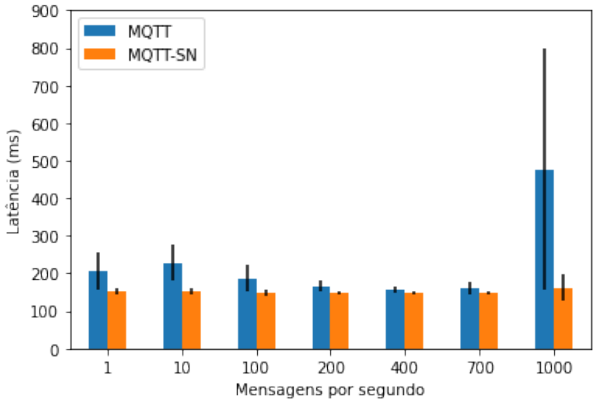


**Figura 7:** Latência: cenário 4 (8 P : 1 A)

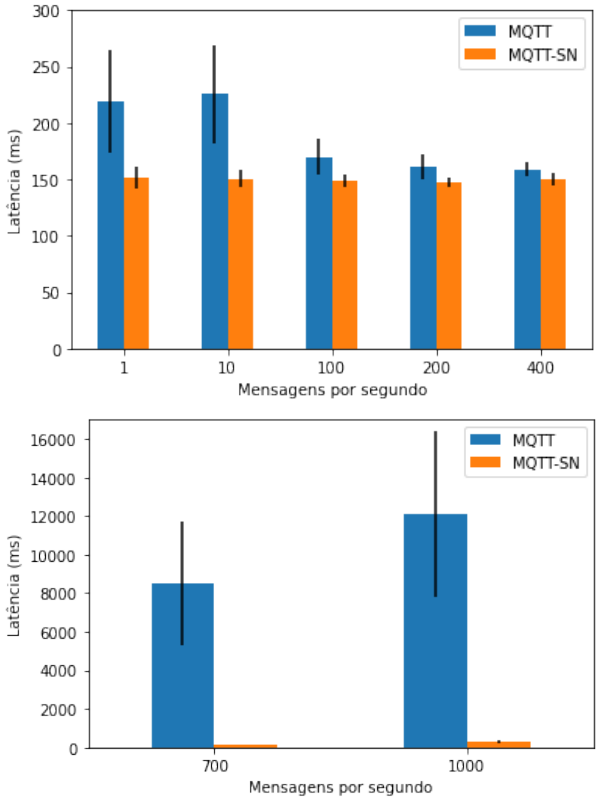


**Figura 8:** Latência: cenário 5 (16 P : 1 A)

situação de um provável gargalo para o protocolo MQTT (latência média de 475 ± 321 ms) na taxa de 1000 msg/s.

Com 16 publicadores (Figs. 8 e 9), observa-se atrasos acentuados na comunicação por parte do MQTT, com resultados de 8535 ± 3195 ms e 12115 ± 4274 ms para 700 msg/s e 1000 msg/s respectivamente. MQTT-SN sofreu menores impactos na latência mas, em contrapartida, apresentou uma média de 42,73% de pacotes perdidos para 1000 msg/s (as perdas registradas pelo MQTT foram insignificantes).

A partir do sexto cenário (Fig. 10), avalia-se o comportamento para múltiplos assinantes e apenas um publicador. Este cenário é análogo ao segundo, apresentando resulta-

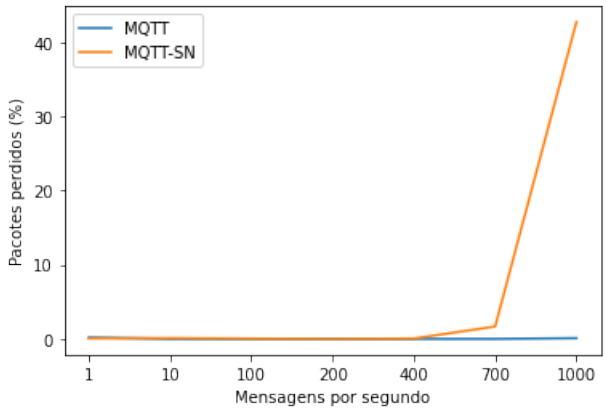


**Figura 9:** Perda de pacotes: cenário 5 (16 P : 1 A)

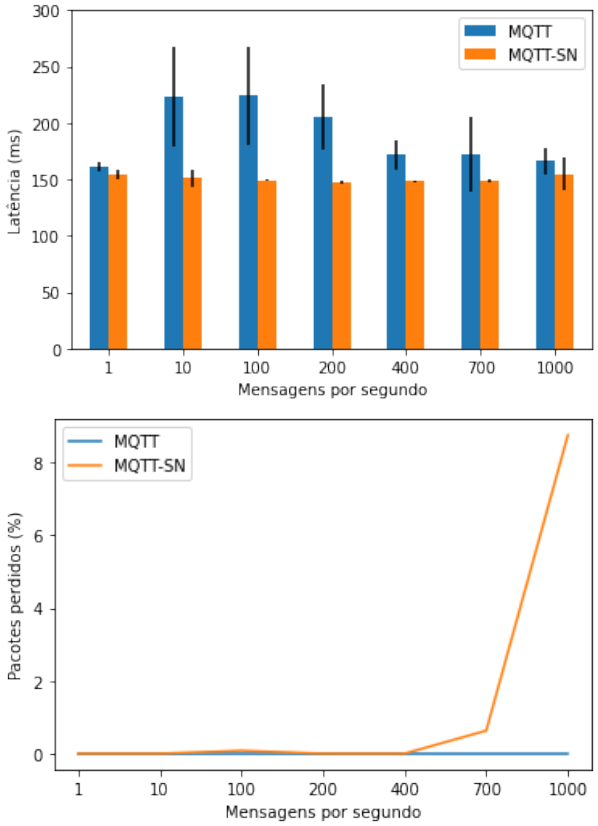


**Figura 11:** Latência: cenário 9 (1 P : 16 A)

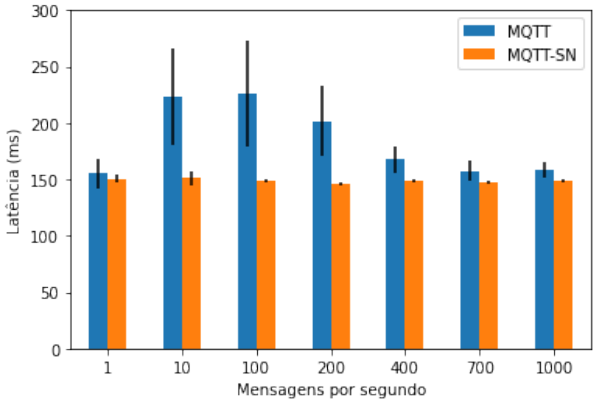


**Figura 10:** Latência: cenário 6 (1 P : 2 A)

dos semelhantes. Os resultados para os cenários sete e oito foram omitidos pois apresentam comportamento muito semelhante ao sexto.

Contrapondo-se aos resultados do quinto cenário (onde há 16 publicadores ao invés de 16 assinantes), no nono cenário (Fig. 11) os atrasos são menores e mais equilibrados, bem como se nota uma redução na perda de pacotes no MQTT-SN (i.e., um máximo de 8,75% para 1000 msg/s, contra mais de 42% no quinto cenário).

No primeiro cenário usando cardinalidade N:N e, mesmo com mais carga, o fluxo de mensagens ainda não é intenso (Fig. 12). Com intervalo de 4 msg/s até 4 mil msg/s não apresentou nenhuma forma de gargalo.

No cenário 11 (4:4), observa-se uma maior simetria na latência entre os dois protocolos (Fig. 13).

No cenário 12 (Figs. 14 e 15), identifica-se um gargalo na comunicação via MQTT-SN. Com 200 msg/s, ele teve 4,19% de pacotes perdidos. Para 400, 700 e 1000 msg/s ocorreram 49,06%, 72,98% e 90,92% de perdas respectivamente. Em contrapartida, o MQTT apresentou resultados de latência consideravelmente superiores ao MQTT-SN (atingindo até 1352 ms com 700 msg/s), mas sem perdas significativas de pacotes.

O cenário 13 apresenta a maior demanda de recursos dentre todos os cenários, evidenciando claramente a existência de gargalos (Figs. 16 e 17). Isso pode ser pontuado tanto com a latência quanto com a perda de pacotes. O MQTT apresenta um típico aumento na latência para evi-

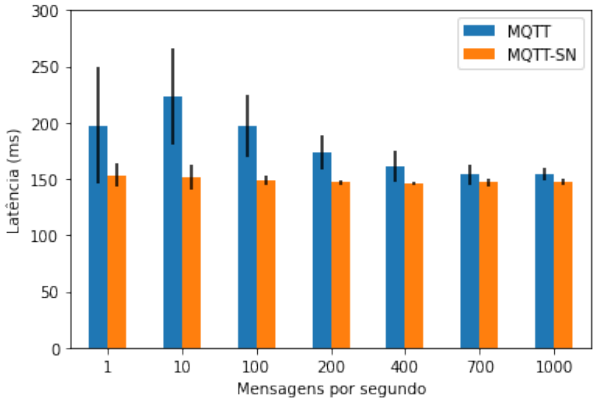


**Figura 12:** Latência: cenário 10 (2 P : 2 A)

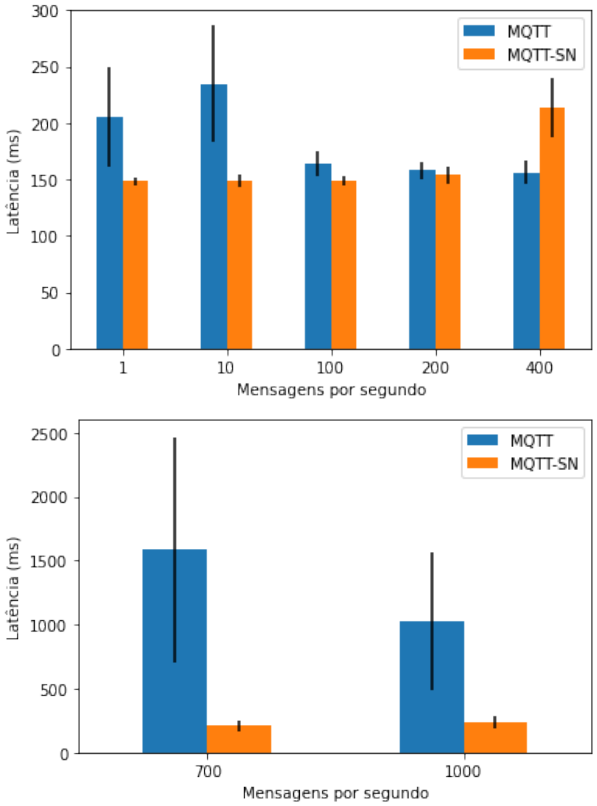


**Figura 14:** Latência: cenário 12 (8 P : 8 A)

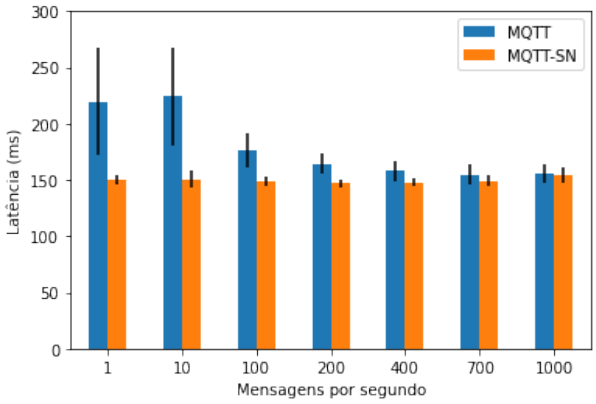


**Figura 13:** Latência: cenário 11 (4 P : 4 A)

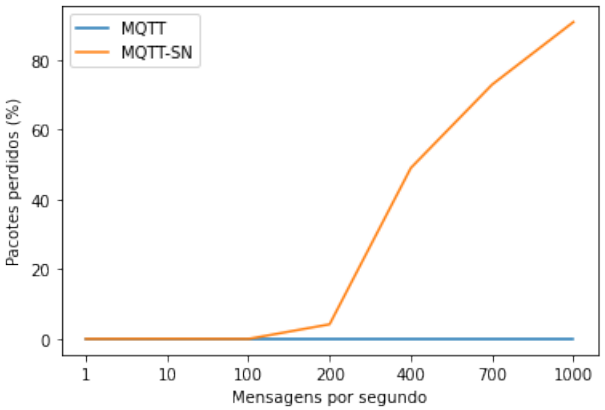


**Figura 15:** Perda de pacotes: cenário 12 (8 P : 8 A)

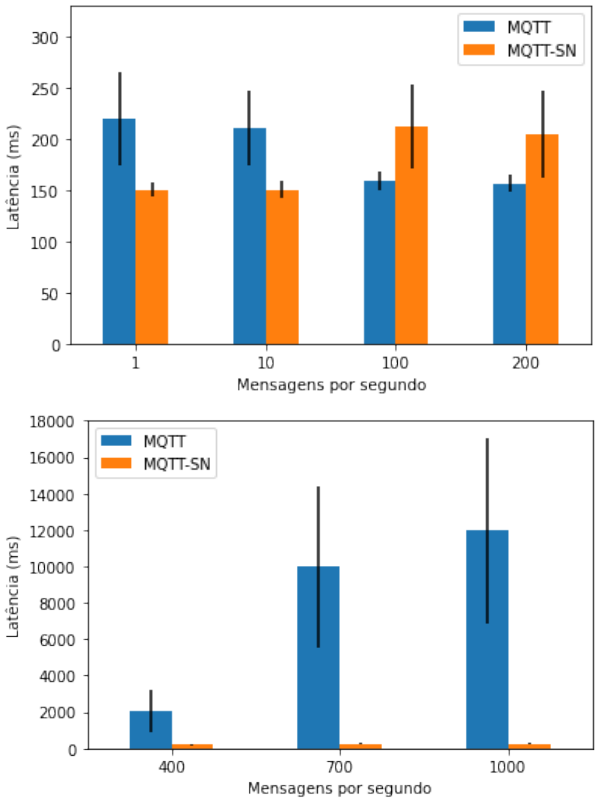


**Figura 16:** Latência: cenário 13 (16 P : 16 A)

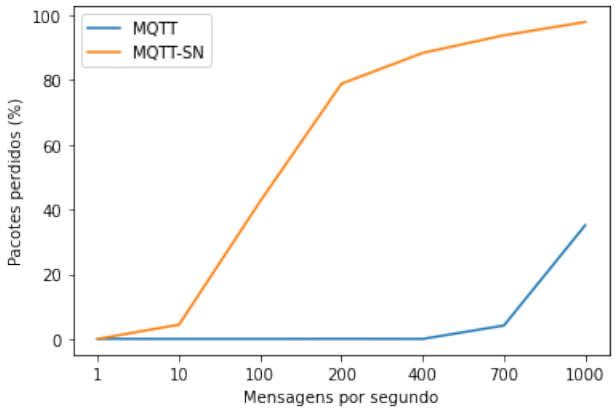


**Figura 17:** Perda de pacotes: cenário 13 (16 P : 16 A)

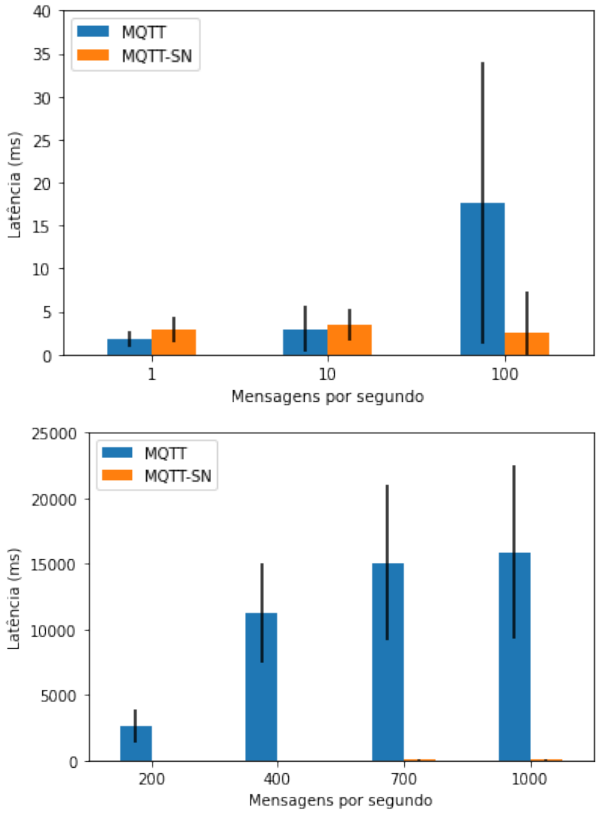


**Figura 18:** Latência: cenário 14 (16 P : 16 A)

tar perdas de pacotes, apesar de não ser tão efetivo para 1000 msg/s onde se observa 35,13% de perda de pacotes. Por sua vez, o MQTT-SN chegou a registrar 254,11 ms de latência média máxima para 700 msg/s. No entanto, a perda de pacotes chegou a 97,9% no pior caso (1000 msg/s).

Esse cenário difere dos demais, com os clientes instanciados em uma máquina AWS EC2 tendo as mesmas características e região do *broker*. Os demais parâmetros são idênticos ao cenário 13. Ao contrário de todos os cenários anteriores, onde a máquina local se constitui um fator de potencial gargalo, observa-se resultados evidenciando novamente a situação de impedimento do sistema em corresponder à demanda gerada pelos clientes, sobretudo quando aplicado o protocolo MQTT-SN.

## 5.1 Discussão

Os resultados de desempenho apresentados na seção anterior evidenciam as diferenças entre os protocolos, principalmente na forma como eles reagem nas situações de gargalo. O MQTT tem como benefício o suporte à retransmissão de mensagens oferecido pelo protocolo TCP. As retransmissões demandam mais recursos, impactando na latência. Por sua vez, o MQTT-SN, sendo baseado no UDP, está sujeito a prática do melhor esforço, sem o recurso de retransmissão de mensagens perdidas. Apesar de oferecer uma latência inferior em alguns momentos, tem-se um percentual bem reduzido de pacotes entregues.

Em situações de poucas mensagens sendo trafegadas

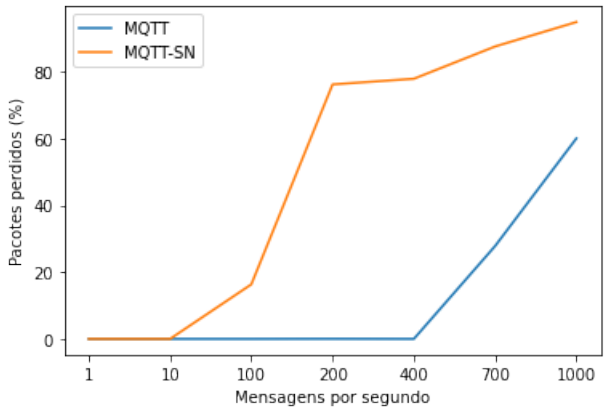


**Figura 19:** Perda de pacotes: cenário 14 (16 P : 16 A)

(*e.g.*, 1, 10, 100), independente da quantidade de clientes, a latência foi próxima ao *Round Trip Time*, e as perdas nulas ou insignificantes. Ao inverter a cardinalidade entre publicadores e assinantes, observa-se atrasos/perdas maiores ao incrementar o número de publicadores comparativamente ao acréscimo de assinantes. Isso ocorre porque, com múltiplos assinantes, as mensagens são replicadas apenas no *broker*, diminuindo assim a chance de perdas no caminho entre os publicadores e o servidor.

Os resultados obtidos são semelhantes aos apresentados por Chen and Kunz (2016), onde o MQTT apresenta latência crescente para incrementos no quantitativo de clientes. A suscetibilidade à perda de pacotes do MQTT-SN reforça o estudo de Cosmi and Mota (2019), onde se destaca a confiabilidade de entrega como um ponto negativo ao protocolo. De forma análoga ao nosso trabalho, considerou-se apenas a qualidade de serviço zero em decorrência de sua reduzida carga de controle.

Há um conjunto restrito de alternativas para avaliação do MQTT-SN. Apesar do *mqtt-sn-tools* (Humfrey, 2013) ser destacado como limitado no estudo de Cosmi and Mota (2019), provou-se adequado para as execuções realizadas neste trabalho. As recomendações apresentadas no estudo de Mun et al. (2016) para configuração de ambientes de teste na nuvem também foram bem aproveitadas em nossas execuções.

## 6 Conclusões

Do ponto de vista do projetista e desenvolvedor, a principal vantagem de um protocolo como o MQTT-SN está na manutenção da compatibilidade com o MQTT. O emprego do UDP como protocolo de transporte requer a existência de uma ponte entre a nova variante e o *broker* MQTT; papel desempenhado pelo *gateway*. Este, por sua vez, geralmente fica hospedado na borda da infraestrutura, ou próximo dela. O emprego de um protocolo de transporte de melhor esforço permite reduzir o número agregado de transmissões pelos dispositivos, adequando-se a canais de largura de banda estreita. No entanto, perdas devido a erros de transmissão, ou resultantes de congestionamento, podem ter consequências inaceitáveis a depender da informação perdida.

A amplitude dos testes de desempenho apresentados nesse trabalho permitem identificar os pontos limites de cada um dos protocolos. Assim como nos trabalhos relacionados, focou-se no nível de qualidade de serviço de melhor esforço, objetivando-se analisar o comportamento dos protocolos em situações de esgotamento de recursos. O MQTT-SN apresenta o pior resultado para situações de alta taxa de transmissão de mensagens, em decorrência dos efeitos colaterais do UDP. O MQTT, por utilizar o TCP, contorna situações de perda de pacotes com o mecanismo de retransmissão do protocolo subjacente. Em contrapartida, o número de retransmissões impacta diretamente no atraso médio de transmissão das mensagens. Neste caso, a depender da aplicação, é possível que a extrapolação no atraso impossibilite a execução de uma aplicação mesmo que não ocorra a ausência de mensagens ao cliente final. Enquanto a qualidade de serviço de melhor esforço tem consequências perceptíveis ao MQTT apenas em momentos de desconexão do cliente, no MQTT-SN é um fator de atenção contínuo em decorrência do emprego do UDP.

Para trabalhos futuros, tem-se uma quantidade fatorial de combinações de parâmetros de configuração de ambos os protocolos para serem explorados. Dentre estes, elenca-se a combinação dos diversos níveis de qualidade de serviço. Além disso, ampliar os possíveis cenários de infraestrutura da aplicação, permitindo também avaliar o desempenho em outras situações que impactem na perda de pacotes (e.g., erros de transmissão em canais via radiofrequência com largura de banda estreita).

## Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications, *IEEE Communications Surveys Tutorials* **17**(4): 2347–2376. https://doi.org/10.1109/COMST.2015.2444095.

Amaran, M. H., Noh, N. A. M., Rohmad, M. S. and Hashim, H. (2015). A comparison of lightweight communication protocols in robotic applications, *Procedia Computer Science* **76**: 400–405. https://doi.org/10.1016/j.procs.2015.12.318.

Aures, G. and Lübben, C. (2019). Dds vs. mqtt vs. vsl for iot, *Network* **1**. https://doi.org/10.2313/NET-2019-10-1_01.

Banks, A., Briggs, E., Borgendale, K. and Gupta, R. (2019). *MQTT v5.0 standard*, OASIS. Disponível em http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

Chaudhary, A., Peddoju, S. K. and Kadarla, K. (2017). Study of internet-of-things messaging protocols used for exchanging data with external sources, *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, IEEE, pp. 666–671. https://doi.org/10.1109/MASS.2017.85.

Chen, Y. and Kunz, T. (2016). Performance evaluation of iot protocols under a constrained wireless access network, *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, IEEE, pp. 1–7. https://doi.org/10.1109/MoWNet.2016.7496622.

Cosmi, A. B. and Mota, V. F. (2019). Uma análise dos protocolos de comunicação para internet das coisas, *Anais do III Workshop de Computação Urbana*, SBC, pp. 153–166. https://doi.org/10.5753/courb.2019.7475.

Eclipse, F. (2013). Rsmb: Really small message broker. Disponível em https://github.com/eclipse/mosquitto.rsmb.

Eclipse, F. (2016). Eclipse paho mqtt go client. Disponível em https://github.com/eclipse/paho.mqtt.golang.

Foundation, E. (2021). Eclipse mosquitto: An open source mqtt broker. Disponível em https://mosquitto.org.

Humfrey, N. (2013). Mqtt-sn tools. Disponível em https://github.com/njh/mqtt-sn-tools.

Hunkeler, U., Truong, H. L. and Stanford-Clark, A. (2008). Mqtt-s—a publish/subscribe protocol for wireless sensor networks, *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, IEEE, pp. 791–798. https://doi.org/10.1109/COMSWA.2008.4554519.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F. and Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things, *Transaction on IoT and Cloud computing* **3**(1): 11–17. Disponível em https://www.researchgate.net/publication/303192188_A_survey_on_application_layer_protocols_for_the_Internet_of_Things.

Mun, D.-H., Le Dinh, M. and Kwon, Y.-W. (2016). An assessment of internet of things protocols for resource-constrained applications, *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1, IEEE, pp. 555–560. https://doi.org/10.1109/COMPSAC.2016.51.

Naik, N. (2017). Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http, *2017 IEEE international systems engineering symposium (ISSE)*, IEEE, pp. 1–7. https://doi.org/10.1109/SysEng.2017.8088251.

Pardo-Castellote, G. (2003). Omg data-distribution service: Architectural overview, *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, IEEE, pp. 200–206. https://doi.org/10.1109/MILCOM.2003.1290110.

Quincozes, S., Emilio, T. and Kazienko, J. (2019). Mqtt protocol: Fundamentals, tools and future directions, *IEEE Latin America Transactions* **17**(09): 1439–1448. https://doi.org/10.1109/TLA.2019.8931137.

Santos, B. P., Silva, L. A., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N. and Loureiro, A. (2016). Internet das coisas: da teoria à prática, *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos* **31**. Disponível em https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf.

Schütz, B., Bauer, J. and Aschenbruck, N. (2017). Improving energy efficiency of mqtt-sn in lossy environments using seed-based network coding, *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, IEEE, pp. 286–293. https://doi.org/10.1109/LCN.2017.87.

Silva, E. F., Dembogurski, B. J., Vieira, A. B. and Ferreira, F. H. C. (2019). Ieee p21451-1-7: Providing more efficient network services over mqtt-sn, *2019 IEEE Sensors Applications Symposium (SAS)*, IEEE, pp. 1–5. https://doi.org/10.1109/SAS.2019.8706107.

Standard, O. (2020). Mqtt version 5.0. Disponível em http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

Subramoni, H., Marsh, G., Narravula, S., Lai, P. and Panda, D. K. (2008). Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (amqp) over infiniband, *2008 Workshop on High Performance Computational Finance*, IEEE, pp. 1–8. https://doi.org/10.1109/WHPCF.2008.4745404.

Talaminos-Barroso, A., Estudillo-Valderrama, M. A., Roa, L. M., Reina-Tosina, J. and Ortega-Ruiz, F. (2016). A machine-to-machine protocol benchmark for ehealth applications–use case: Respiratory rehabilitation, *Computer methods and programs in biomedicine* **129**: 1–11. https://doi.org/10.1016/j.cmpb.2016.03.004.

Viriato, V., Martin and Conceicao, J. D. (n.d.). Introduction to mqtt-sn (mqtt for sensor networks). Disponível em http://www.steves-internet-guide.com/mqtt-sn/.

Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S. and Al-Hatmi, R. (2017). Internet of things: Survey and open issues of mqtt protocol, *2017 International Conference on Engineering & MIS (ICEMIS)*, IEEE, pp. 1–6. https://doi.org/10.1109/ICEMIS.2017.8273112.