COMPARAÇÃO DE PROTOCOLOS

Hamilton Luiz Bez Batti Dias, Gabriel Medeiros Gomes da Silva
Discentes da disciplina de Sistemas Ubíquos
Araranguá, SC, Brasil - Universidade Federal de Santa Catarina - DEC
Emails: hamiltondias47@gmail.com, gabrielmedeiros0605@gmail.com

RESUMO: Aplicações para a Internet das Coisas são cada vez mais requisitadas à medida que o mundo se globaliza. Com isso, é importante escolher o protocolo certo para comunicar as máquinas. O presente artigo relata como os protocolos da camada de aplicação MQTT-SN (*Message Queuing Telemetry Transport for Sensor Networks)* e CoAP (*Constrained Application Protocol*) foram comparados. Foi visto que esses protocolos não foram testados para as métricas utilizadas no presente texto. O objetivo foi contribuir, de forma analítica e prática, para as análises de protocolos para Internet das Coisas. Para tanto, foram implementados ambos os protocolos em sistemas embarcados e realizados testes de vazão, perda de pacotes, latência e tempo de transmissão sobre tamanho da mensagem. Considerando apenas uma comunicação ponto a ponto, apenas cliente/cliente passando por um servidor/*broker*. Após, com os resultados, percebe-se que o CoAP é um protocolo que tem menos perda de pacotes, mas sua latência e vazão são inferiores quando comparado com o MQTT-SN. Por fim, cita-se aplicações para ambos protocolos.

PALAVRAS CHAVE: Internet das Coisas. MQTT-SN. CoAP. Comparação.

ABSTRACT: Applications for Internet of Things are increasingly on demand as the world is more globalized. It is important to choose the right protocol to machine to machine communication. This article reports how protocols from the application layer MQTT-SN (Message Queuing Telemetry Transport for Sensor Networks) and CoAP (Constrained Application Protocol) were compared. The objective was to contribute, by analytical and practical means, to protocols analysis for Internet of Things. It was seen that those two protocols weren't tested for the metrics in this article. Therefore, both protocols were implemented in embedded systems and tested for flow rate, packet loss, latency and transmission time over message size. Considering only peer to peer communication, only client/client going through a server/broker. After, with the results, one notices that CoAP is a protocol that has less packets losses, but It's latency and flow rate are inferior when compared to MQTT-SN. In the end, are quoted application for both protocols.

KEY-WORDS: Internet of Things. MQTT-SN. CoAP. Comparison.

1 INTRODUÇÃO

A tecnologia antes de ser como é conhecida hoje, passou por 4 (quatro) grandes mudanças, que são: tecnologias eletromecânicas, tecnologias de difusão, tecnologias do disponível e os computadores pessoais. O primeiro é trouxe a reprodutibilidade técnica, onde surgiram as linhas de produções de jornal, câmeras fotográficas e outras tecnologias, mas esses mecanismos não eram programáveis. A segunda era trouxe tecnologias de transmissão a baixo custo ao público. As tecnologias do disponível possibilitou que as pessoas pudessem buscar conteúdos a sua escolha, através de vídeo cassete, *walkman* e televisão a cabo. Com a quarta era computadores se tornam populares e em um curto período de tempo, os computadores e celulares começam a se tornar a mesma coisa (SANTAELLA, L.; GALA, A.; POLICARPO, C.; GAZONI, R. 2013).

Com a possibilidade de criar *hardware* baratos e programáveis, surge uma nova ideia da computação: a Internet das Coisas (*Internet of Things,* IoT), que introduz uma quebra de paradigma em várias áreas. A IoT prevê um mundo onde geladeiras, carros, estoques e praticamente todas as outras “coisas” estarão conectadas à Internet. (BEN-DAYA, M.; HASSINI, E.; BAHROUN, Z., 2017; XIA, Feng *et al*., 2012).

A Internet das Coisas vai aumentar a onipresença da Internet por integrar cada objeto do cotidiano através de um sistema embarcado, que trás uma rede distribuída de dispositivos que se comunicam com humanos e também entre si (XIA, Feng *et al*., 2012).

De acordo com AGRAWAL, S. & VIEIRA, D. o ambiente que receberá a maior mudança com a IoT é o ambiente doméstico, como as *smart houses* (casas inteligentes), *smart cars* (carros inteligentes) e as mais variadas assistências. A qualidade de vida pessoal do ser humano deve aumentar significamente. Do ponto de vista profissional, a IoT deve trazer maior qualidade dos produtos, melhor e mais controlada produção e melhores serviços em geral.

Junto com as novas tecnologias, vêm novos desafios. Além dos desafios já citados, também vale ressaltar a eficiência energética e ambientes limitados (largura

de banda, dispositivos etc). Muitos protocolos tentam resolver alguns desses problemas. (MAZZER, D., FRIGIERI, E. P., PEREIRA, L. F. P. P., 2015).

Apesar da grande quantidade de materiais sobre ambos os protocolos, em nenhum artigo foi encontrado a demonstração ou realização de testes de vazão, perda de pacotes, tempo de transmissão por tamanho do pacote e latência.

Esse trabalho tem objetivo apresentar, discutir e analisar duas de tais soluções (protocolos) de comunicação: Message Queuing Telemetry Transport for Sensor Networks (MQTT-SN) e Constrained Application Protocol (CoAP). As métricas para análise serão os quatro tópicos que não foram encontradas na literatura.

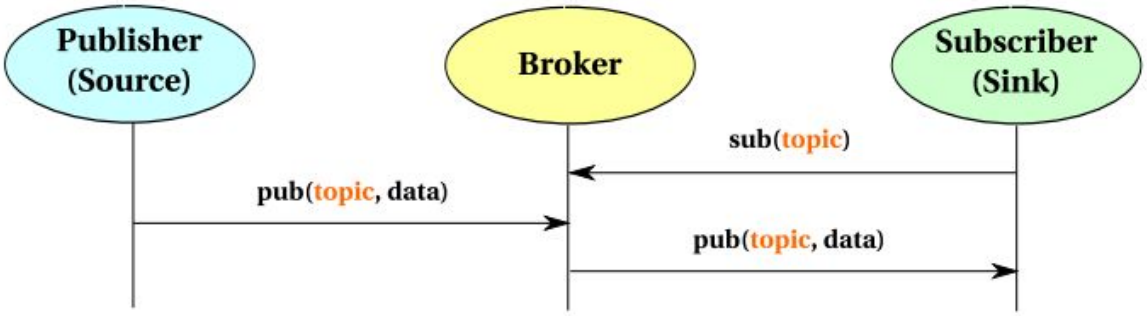
2 MQTT-SN

O MQTT (*Message Queuing Telemetry Transport*) foi desenvolvido pela IBM no fim dos anos 90 para ser um protocolo leve, de fácil utilização, baixo custo e sobre o TCP/IP. No fim de 2014, o MQTT se tornou um padrão aberto OASIS (AMARAN, M. H., *et al.,* 2015)

Em 2013 foi criado o MQTT-S, uma variante do MQTT só que mais focado em dispositivos embarcados, com ainda menos capacidade de processamento e energia. O nome foi mudado de MQTT-S para MQTT-SN (*Message Queuing Telemetry Transport for Sensor Networks*) pois causava confusão, já que as pessoas pensavam que o S significava segurança (AMARAN, M. H., *et al.,* 2015).

Ambos os protocolos funcionam de forma publicador/inscrito (em inglês *publisher/subscriber, pub/sub*), que funciona da seguinte forma: clientes que se interessam por algum tipo de informação se inscrevem para receber tal informação (sendo chamado de inscrito ou *subscriber*), e clientes que produzem um tipo de informação publicam elas (sendo chamados de publicadores, ou *publishers*), e o meio que recebe os dados dos *publishers* e envia para os *subscribers* é chamado de *broker.* O *broker* coordena tudo, desde as inscrições, até as publicações (HUNKELER, U., TRUONG, H., L., STANFORD-CLARK, A., 2008).

Figura 1 - Modelo de Comunicação *pub/sub*

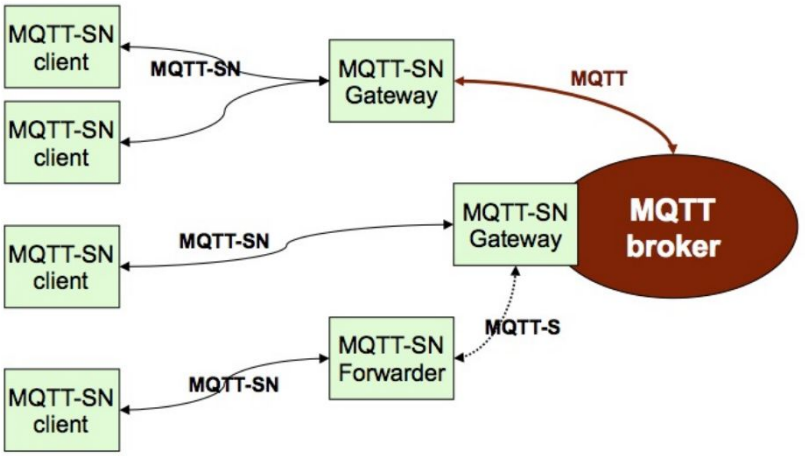


Fonte: HUNKELER, U., TRUONG, H., L., STANFORD-CLARK, A., 2008

A Figura 1 mostra um exemplo de como a comunicação *pub/sub* funciona. Um cliente *subscriber* envia uma mensagem de inscrição para o *broker* que contém o *topic*, o nome do tópico que ele quer se inscrever. O cliente *publisher* publica dados (*dada*) em um tópico específico (nesse caso, *topic*). O *boker* envia os dados recém publicados para todos os clientes inscritos no tópico *topic.*

O MQTT-SN não usa apenas o TCP, ele também pode operar em cima do UDP. Para AMARAN, M. H., *et al.,* a principal vantagem de usar o UDP é que remove os *handshakes* do TCP, fazendo com que a comunicação seja sem conexão e assim, mais leve.

Figura 2 - Arquitetura do MQTT-SN



Fonte: STANFORD-CLARK, A., TRUONG, H. L., 2013.

A arquitetura do MQTT-SN é mostrada na Figura 2. No MQTT-SN é adicionado mais dois componentes à arquitetura *pub/sub*: O *Gateway* e o *Forwarder*. Os clientes sempre precisam se conectar à um *Gateway* com o protocolo MQTT-SN para se comunicar com o *broker*. O *Gateway* pode ou não estar integrado com o *broker*, se não estiver, é feita uma conexão utilizando MQTT. A função do *Gateway* é traduzir MQTT-SN para MQTT e vice-versa. Caso os clientes estejam em uma rede diferente ao *broker*, é necessário um *Forwarder*, que apenas retransmite os pacotes que recebe (sem mudar nada) para o *Gateway* e vice-versa (STANFORD-CLARK, A., TRUONG, H. L., 2013).

3 CoAP

O CoAP (*Constrained Application Protocol*) foi desenvolvido pela ARM e pela IETF (*Internet Engineering Task Force*) e padronizado em 2014. Adaptado do HTTP, foi criado pensando em dispositivos embarcados com potência baixa e processamento limitado. É executado em cima do UDP e utiliza o modelo requisição/resposta (MAZZER, D., FRIGIERI, E. P., PEREIRA, L. F. P. P., 2015 e AMARAN, M. H., *et al.,* 2015).

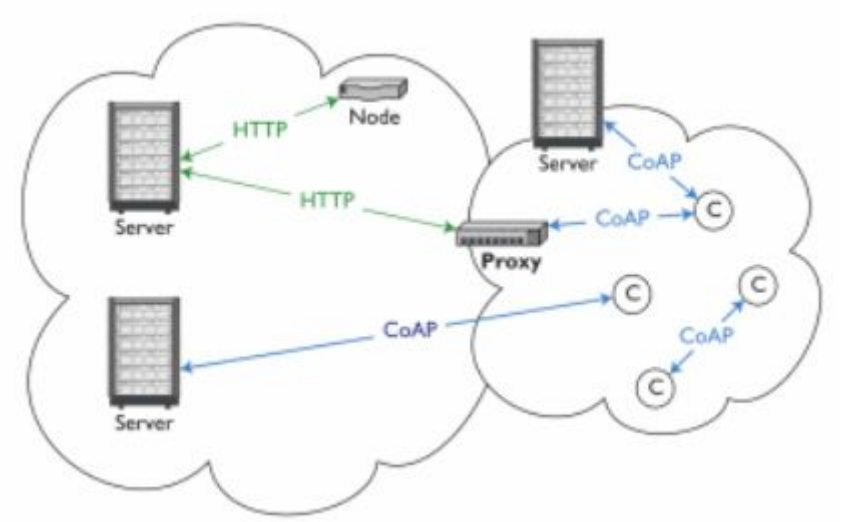
A troca de mensagens é parecida com o HTTP, com a comunicação assíncrona em cima do UDP. O cliente faz requisições de ações ou recursos para o servidor, e o servidor envia a resposta ao cliente (PORCIÚNCULA, C. B. da *et al., 2018*).

Existem quatro tipos de mensagens no CoAP, a CON (confirmável), NON (não confirmável), ACK (reconhecimento) e RESET. A primeira, são mensagens que precisam de confirmação do destino. A segunda, são mensagens que não precisam de confirmação, como por exemplo, quando um sensor envia mensagens. A terceira, são mensagens de confirmação do primeiro tipo. Por último, a RESET, diz que a mensagem CON ou NON teve algum tipo de erro no processamento e não pode ser lida. A segurança do CoAP é feita através do DTLS (*Datagram Transport Layer*

Security), que é baseado no TLS em cima do UDP (PORCIÚNCULA, C. B. da *et al., 2018* e MARTINS, I. R., ZEM, J. L., 2015).

CoAP usa arquitetura REST que é um modelo de comunicação compartilhado com o HTTP. Essa arquitetura faz com que o CoAP seja operável junto ao HTTP sem ser necessário um grande tradutor como mostra a Figura 3 (AMARAN, M. H., *et al.,* 2015 e MARTINS, I. R., ZEM, J. L., 2015).

Figura 3 - Arquitetura CoAP



BORMANN, C., CASTELLANI, A. P., SHELBY, Z., 2012.

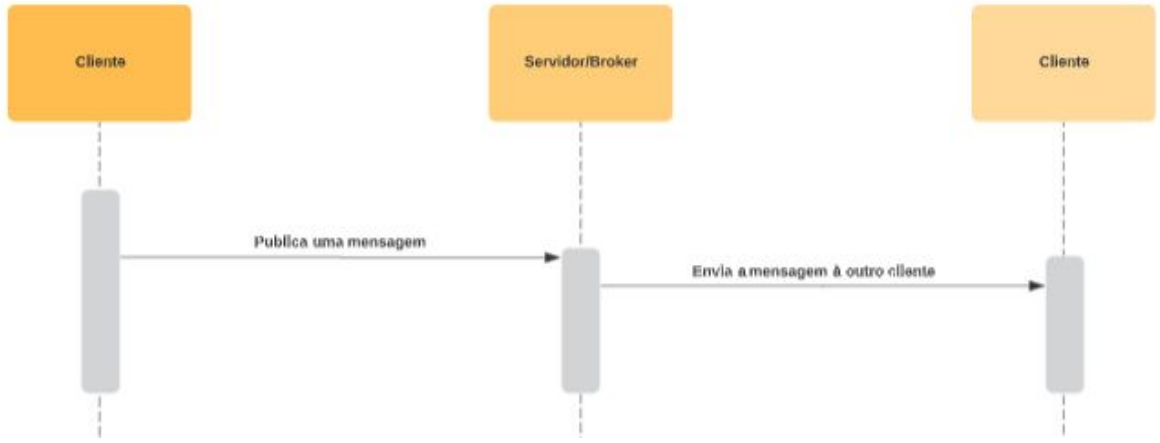
De acordo com a RFC 7252, o CoAP tem as seguintes características:

- Protocolo WEB que cumpre os requerimentos M2M em ambientes restritos;
- Troca de mensagens assíncrona;
- Suporte à URI e *Content-Type;*
- Capacidade simples de *proxy* e *caching.*
- Mapeamento HTTP que permite que *proxies* possam prover acesso aos recursos do CoAP via HTTP de maneira uniforme;
- UDP com confiabilidade opcional;
- *Unicast;*
- *Multicast;*
- Suporte aos métodos GET, POST, PUT e DELETE;

4 METODOLOGIA

Devido à falta de artigos em relação à testes de ambos os protocolos, decidiu-se testar os mesmos em relação à latência, perda de pacotes, vazão e tempo de transmissão da mensagem em relação ao tamanho da mensagem que está sendo enviada. Todos os testes são calculados conforme a mostra a Figura 4, ou seja, um cliente envia um pacote ao servidor, e o servidor encaminha esse pacote a um segundo cliente.

Figura 4 - Arquitetura dos testes



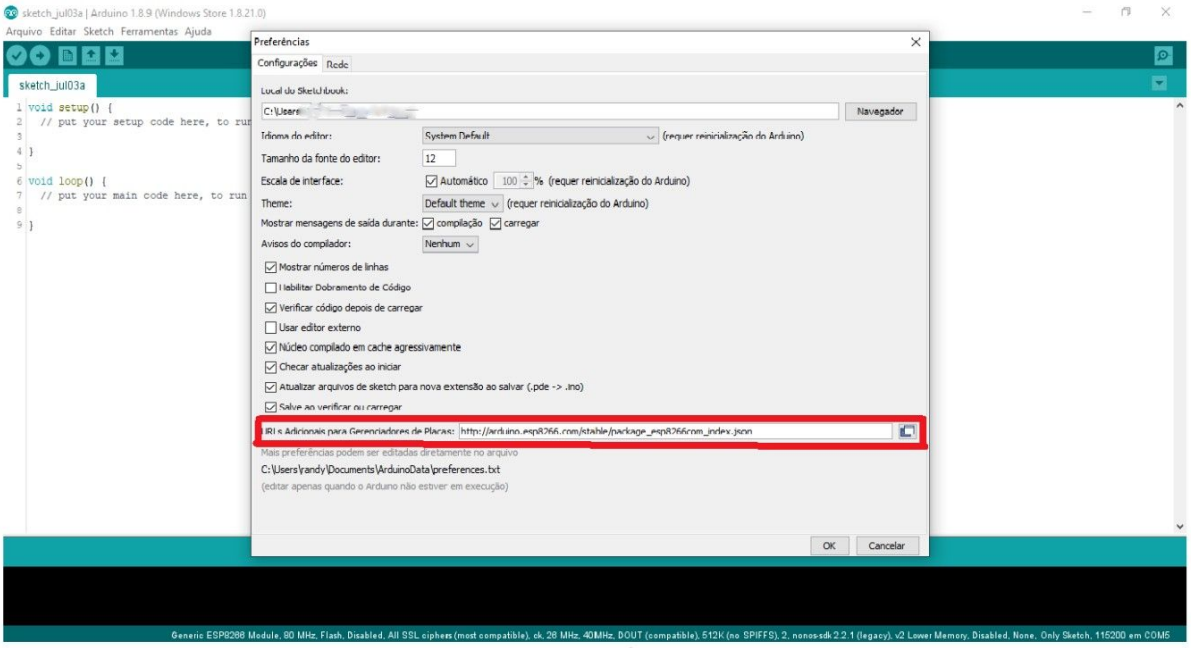
O primeiro teste, de latência, consiste em verificar o tempo que a mensagem leva saindo do primeiro cliente até chegar no segundo cliente. O teste de perda de pacotes, seria para verificar quantos pacotes são perdidos a cada 1000 pacotes enviados. O terceiro teste, de vazão, tem a função de verificar quantos pacotes são enviados em um determinado tempo, e quantos desses pacotes são recebidos pelo outro cliente. O último teste, tem a intenção de verificar se há diferença de tempo, na hora de enviar pacotes com diferentes tamanhos.

Esses dois protocolos foram escolhidos por terem características comuns, como por exemplo, ambos operam sob a camada UDP e tem uma arquitetura do tipo cliente/servidor.

4.1 Implementação do MQTT-SN

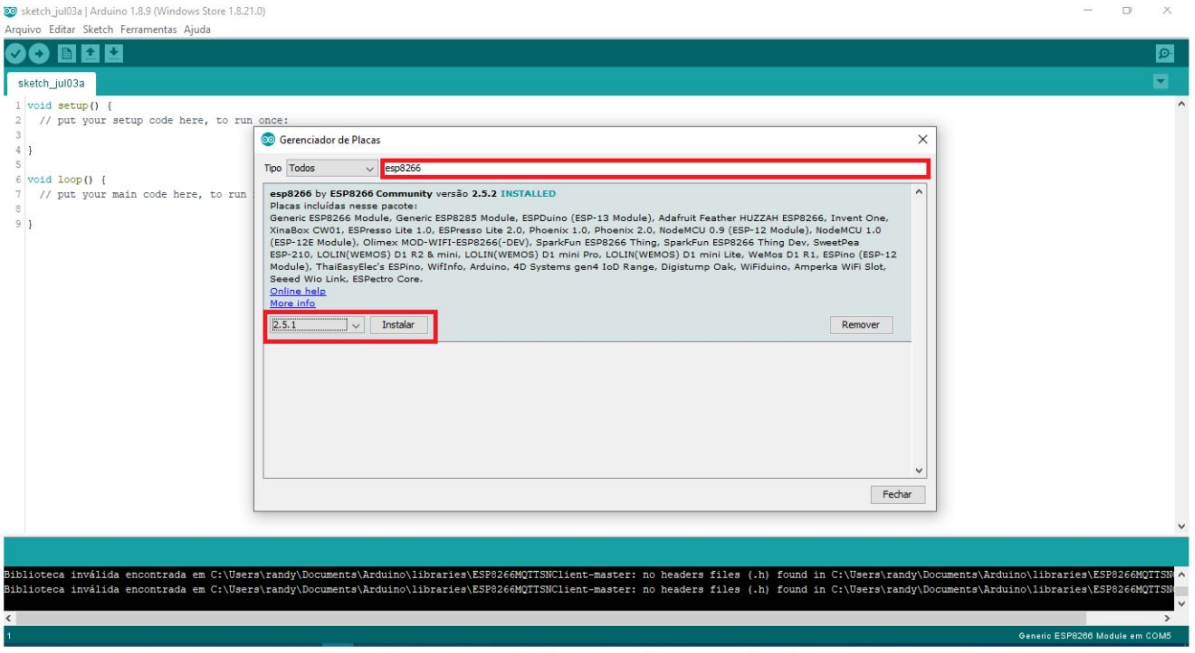
Primeiro, foi baixado a IDE do arduino a partir do link https://www.arduino.cc/en/Main/Software. Após a instalação, adicione o json do esp8266 em Arquivo > Preferências > Configurações > URLs adicionais para gerenciadores de placas > Adicionar a seguinte URL > http://arduino.esp8266.com/stable/package_esp8266com_index.json > OK. Conforme a Figura 5.

Figura 5 - IDE do Arduino



Agora, é necessário instalar a biblioteca da placa seguindo para a aba Ferramentas > Placa > Gerenciamento de Placas > Pesquisar “esp8266” > Selecionar a versão mais recente > Instalar > Fechar. Conforme a Figura 6.

Figura 6 - Instalando a biblioteca da placa



Em seguida, deverá ser alterado a placa na qual a IDE deve compilar o código. No menu Ferramentas > Placas > Selecionar “NodeMCU v1.0” > Fechar.

O próximo passo é instalar a biblioteca do cliente MQTT-SN para sistemas embarcados encontrada no link https://github.com/S3ler/arduino-mqtt-sn-client. Após baixar a biblioteca, a mesma terá que ser extraída para o diretório de bibliotecas do Arduino. A partir de agora, em Arquivo > Exemplos > Arduino mqtt-sn-client > esp8266 terá exemplos de implementações.

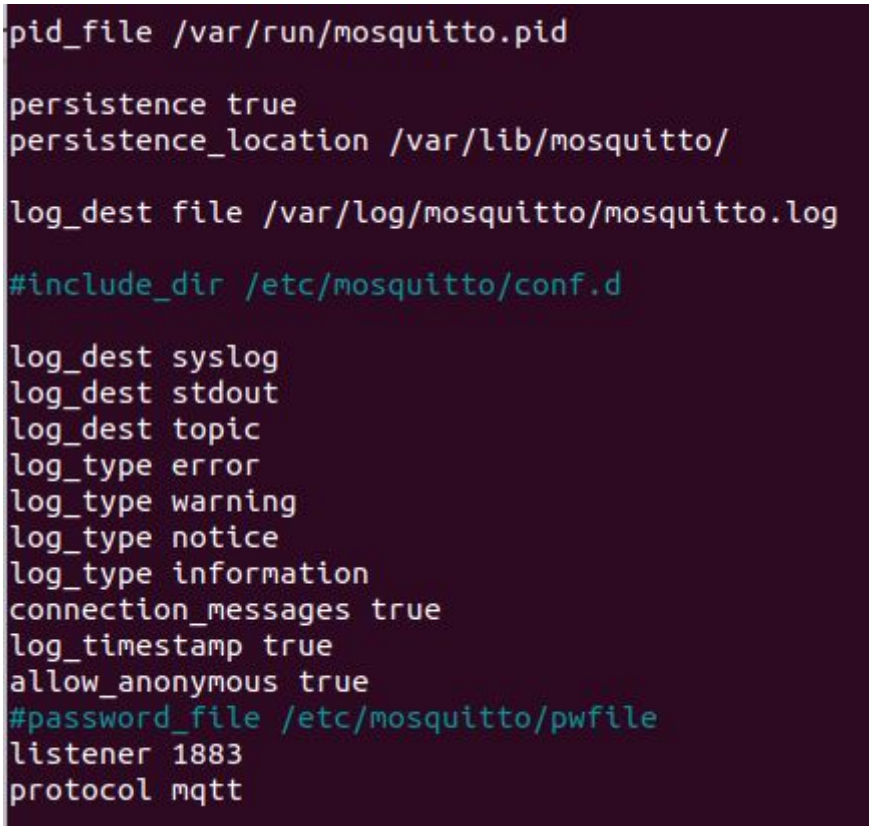
A biblioteca tem apenas dois erros, que são necessários para poder publicar mensagens. Todos os créditos por arrumar esses erros estão no *link* da *issue* (https://github.com/S3ler/arduino-mqtt-sn-client/issues) no *github.* Apenas fazendo as modificações constatadas na issue, e a biblioteca está pronta para uso.

Será utilizado o *Broker* Mosquitto, disponibilizado no seguinte *link:* *https://mosquitto.org/download/*. A maneira mais fácil de instalar no Ubuntu 18.04 LTS é abrir um terminal de comando e digitar “sudo apt-get install mosquitto” e com isso o programa já estará pronto para uso.

Dentro da pasta /etc/mosquitto/ será possível encontrar o arquivo “mosquitto.conf”, que está todas as configurações do *broker*. As configurações

utilizadas nesse artigo estão na Figura 7. Para executar o programa tem que iniciar o programa com o comando: “mosquitto -c /etc/mosquitto/mosquitto.conf”.

Figura 7 - Configuração do *Broker* Mosquitto

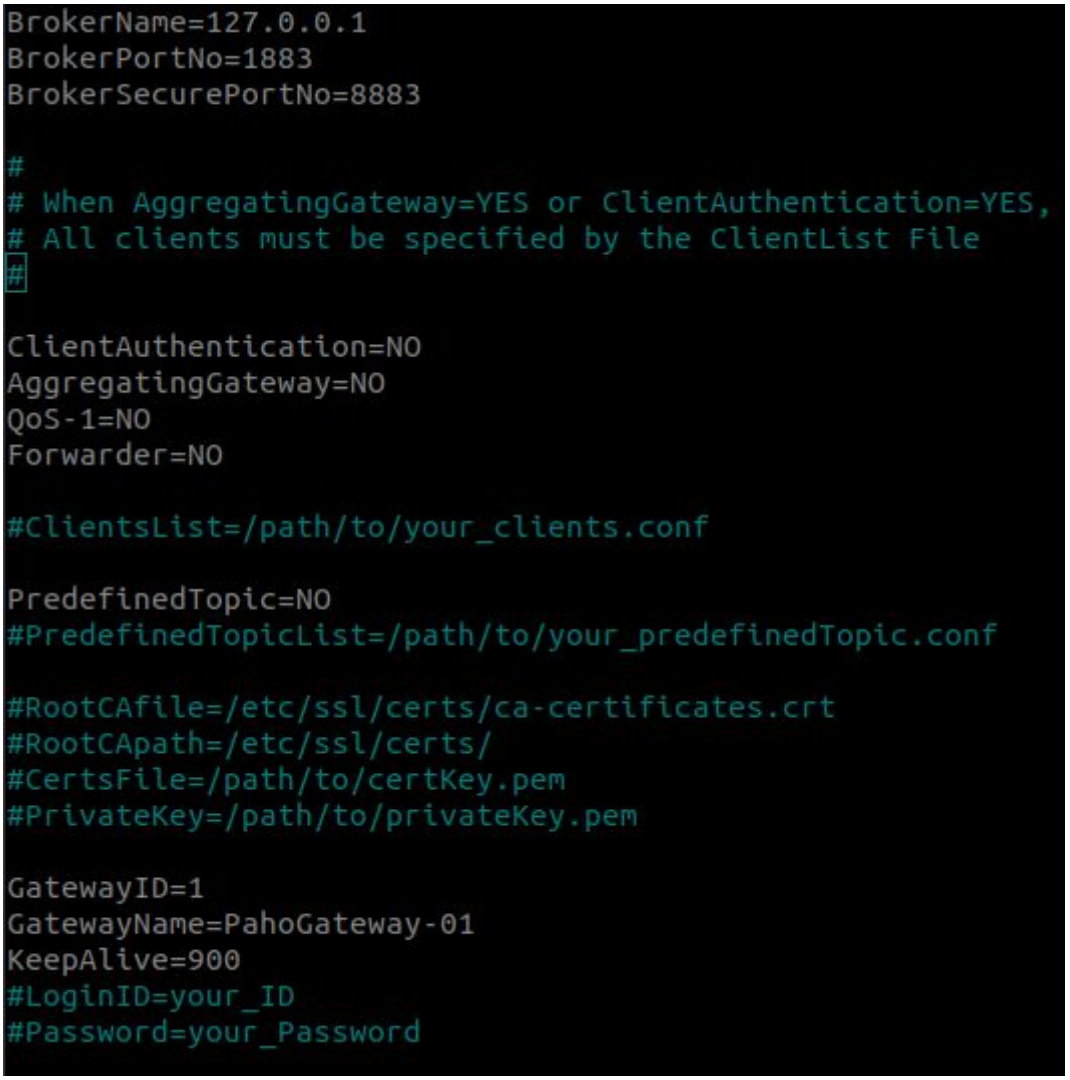


O último passo de preparação é a instalação de um *Gateway.* Será utilizado o Paho Gateway, que está disponível no *link:* *https://github.com/eclipse/paho.mqtt-sn.embedded-c*.

Após baixar e extrair os arquivos, será necessário entrar na pasta “MQTTSNGateway” e executar os seguintes comandos: “make”, “make install” e “make clean”.

Agora, no diretório onde o usuário instalou o seu *gateway,* terá um arquivo chamado “gateway.conf” que consta as configurações do *gateway* e precisa ser modificado. As modificações feitas para esse artigo estão ilustradas na Figura 8.

Figura 8 - gateway.conf



Para iniciar o *gateway,* é necessário apenas executar o seguinte comando: "./MQTTSNGateway" no diretório onde o mesmo foi instalado. Vale ressaltar que o *Gateway* e o *Broker* estão no mesmo computador operando com o Ubuntu 18.04 LTS, enquanto o cliente em um ESP8266.

4.2 Implementação do CoAP

Para a implementação do protocolo CoAP, também foi utilizado a IDE do Arduino, com a biblioteca de desenvolvimento para o ESP8266. Para a instalação dos mesmos, é só seguir os passos mostrados na seção anterior.

Os códigos estão disponíveis no *link* https://github.com/automote/ESP-CoAP. A instalação dessa biblioteca é igual como mostrado na seção anterior.

Com tudo instalado, é só fazer *upload* para dois ESPs (um utilizando o código do cliente, e outro do servidor) e verificar a execução.

5 RESULTADOS

Todos os testes mencionados anteriormente foram feitos em uma região onde apenas se tinha um sinal de Wi-fi, ou seja, não existia interferências de outras redes e nem de outras pessoas.

Durante a realização dos testes foi utilizado a função *millis()* do esp8266, que retorna o tempo em milissegundos desde o *boot* do mesmo. Mas essa função tem dois problemas: 1) 1 milissegundo da função equivale a 1,024 milissegundos reais. Ou seja, esse erro vai se acumulando a cada milissegundo passado. 2) A precisão é de mais ou menos 1 milissegundo, ou seja, quando a função for chamada, a mesma pode ter sido executada 1 milissegundo antes ou depois do valor obtido.

O primeiro problema já é resolvido dentro da própria biblioteca, onde toda vez que esse valor a mais (nesse caso, 0,024) somar 1 milissegundo, o valor do tempo é corrigido. O segundo é um problema de precisão que deve ser considerado em todos os resultados obtidos.

Nos dois protocolos foram utilizados: *payload* mínimo de 1 byte, *payload* máximo de 63 bytes (o MQTT-SN pode ter mais, mas o CoAP pode ter um *payload* máximo de 63 bytes), e *delays* (tempo entre o envio de mensagens) de zero, 0.5 e 1 segundo (acima de um segundo já se obtém o mesmo resultado sempre).

Todos os dados mostrados são médias de vários testes. Os testes são utilizados QoS (*Quality of Service,* ou qualidade de serviço, em português) igual a zero, ou seja, os pacote é enviado pelo cliente e o mesmo não aguarda qualquer tipo de confirmação.

O Gráfico 1 e o Gráfico 2 mostram o resultado obtido do teste de vazão com pacotes pequenos e pacotes grandes, respectivamente. Enquanto que o MQTT-SN no primeiro minuto envia um pouco mais de 5000 pacotes, o CoAP envia

aproximadamente a metade. Mas ambos os protocolos convergem para um valor muito próximo, o de 29064 pacotes.

Gráfico 1 - Comparação de vazão com pacotes pequenos

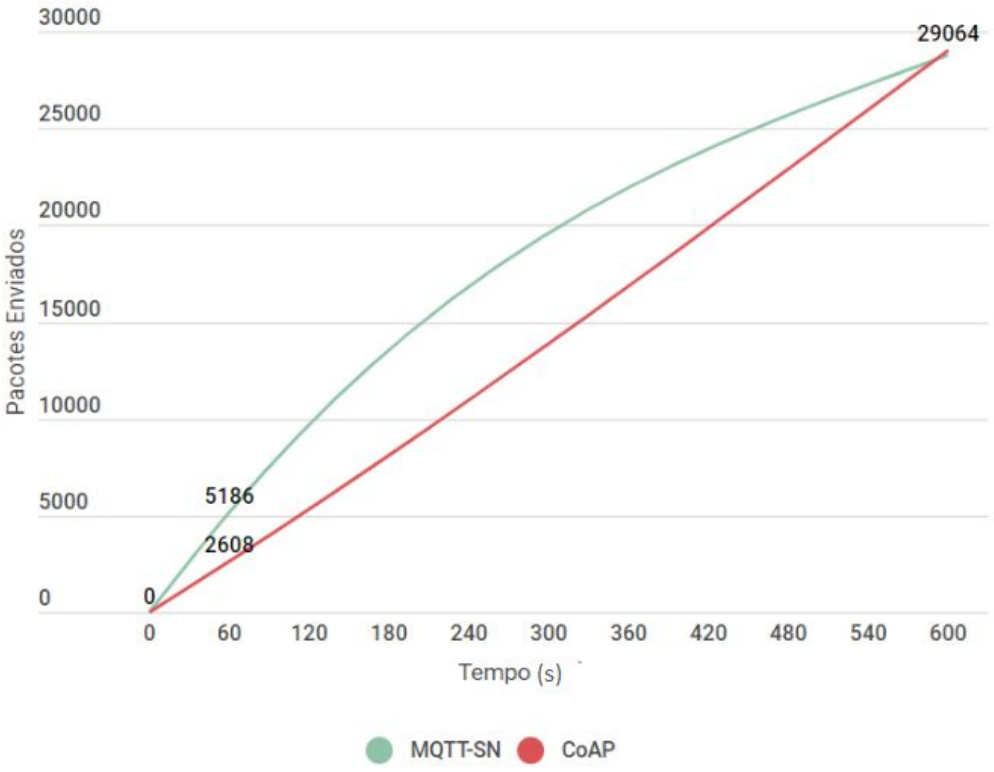
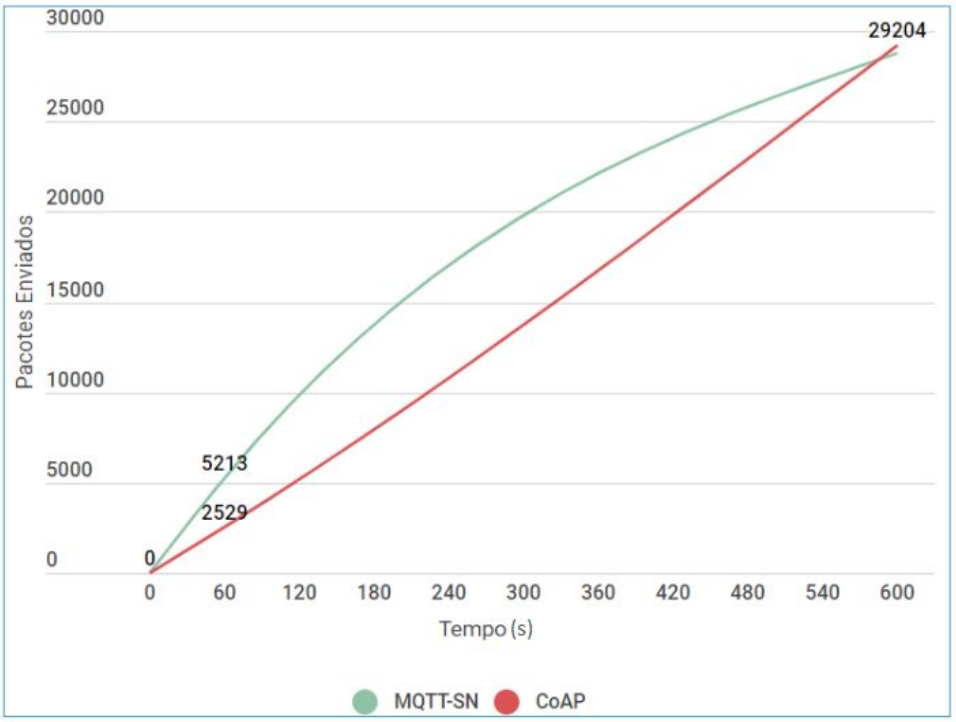


Gráfico 2 - Comparação de vazão com pacotes grandes



O Gráfico 3 e o Gráfico 4 ilustram o resultado obtido de latência com pacotes pequenos e pacotes grandes, respectivamente. A latência do MQTT-SN sem *delay* é dez vezes menor com pacotes pequenos, e essa diferença é ainda maior para pacotes grandes. Mas, como pode ser visto em ambos os gráficos, a latência de ambos fica muito baixa (aproximadamente 1 milissegundo) quando é adicionado um intervalo entre o envio das mensagens.

Gráfico 3 - Comparação de latência com pacotes pequenos

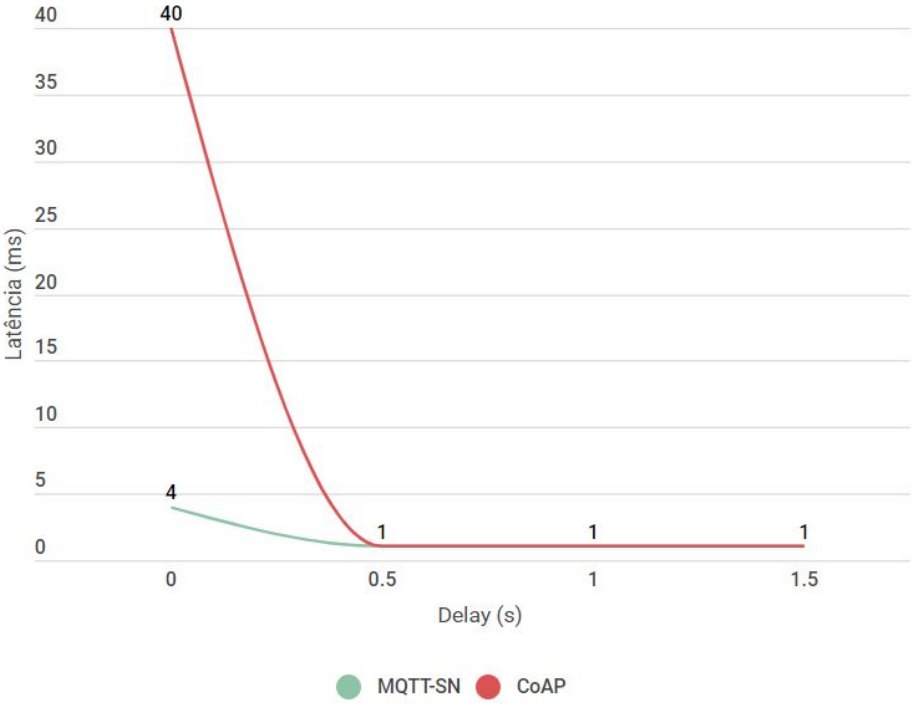
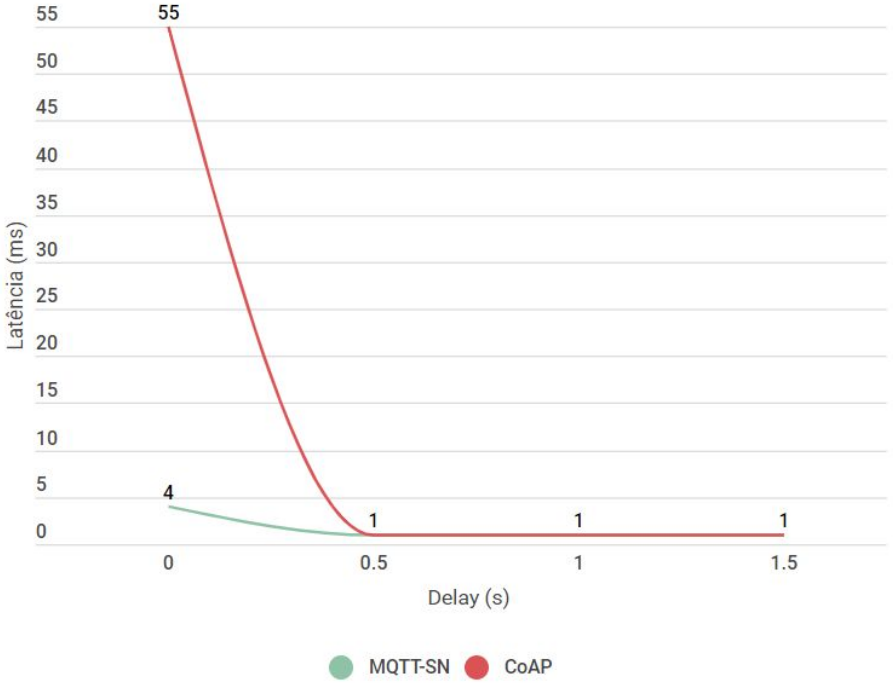


Gráfico 4 - Comparação de latência com pacotes grandes



O Gráfico 5 e o Gráfico 6 ilustram o resultado obtido do experimento de perda de pacotes com pacotes pequenos e pacotes grandes, respectivamente. Nessa métrica, pode-se observar que o CoAP tem um desempenho muito melhor que o MQTT-SN, independente do *delay* entre envios de mensagens. O CoAP observou-se ser um protocolo com mais consistência no envio das mensagens.

Gráfico 5 - Comparação de pacotes perdidos com pacotes grandes

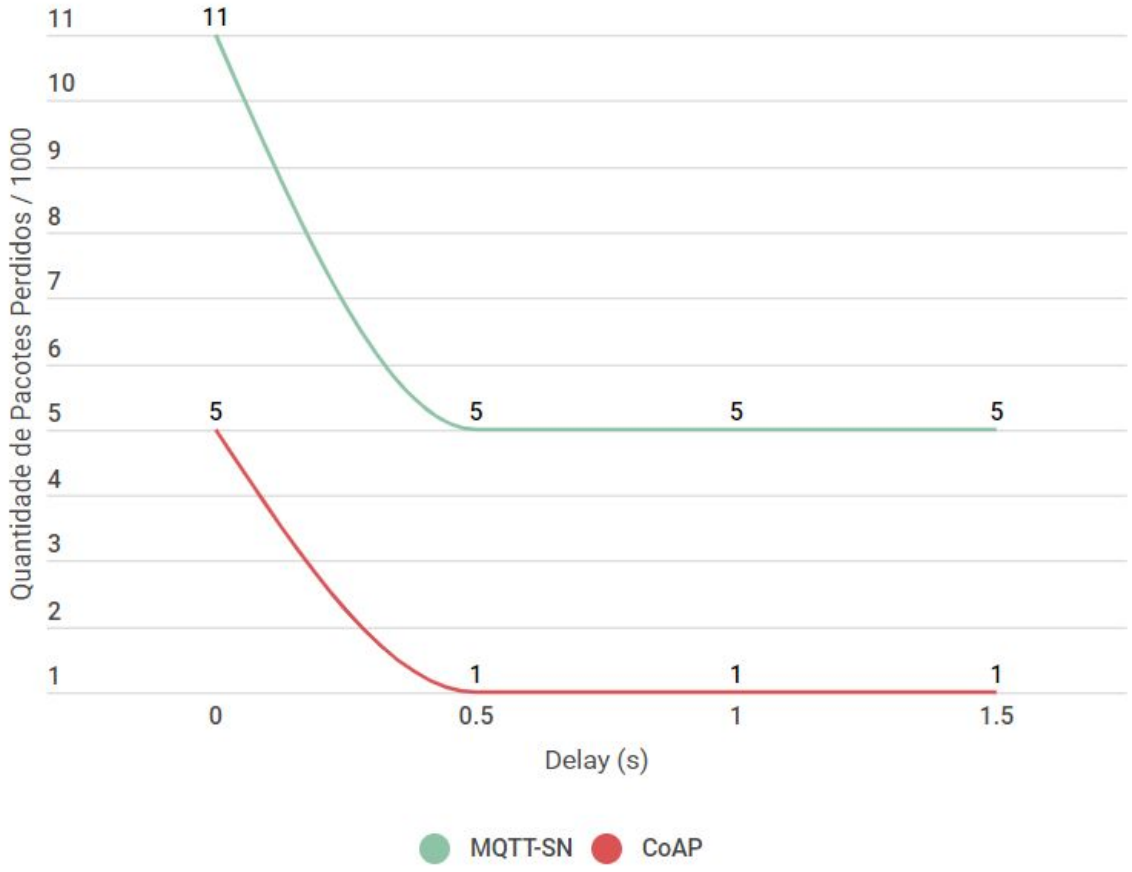
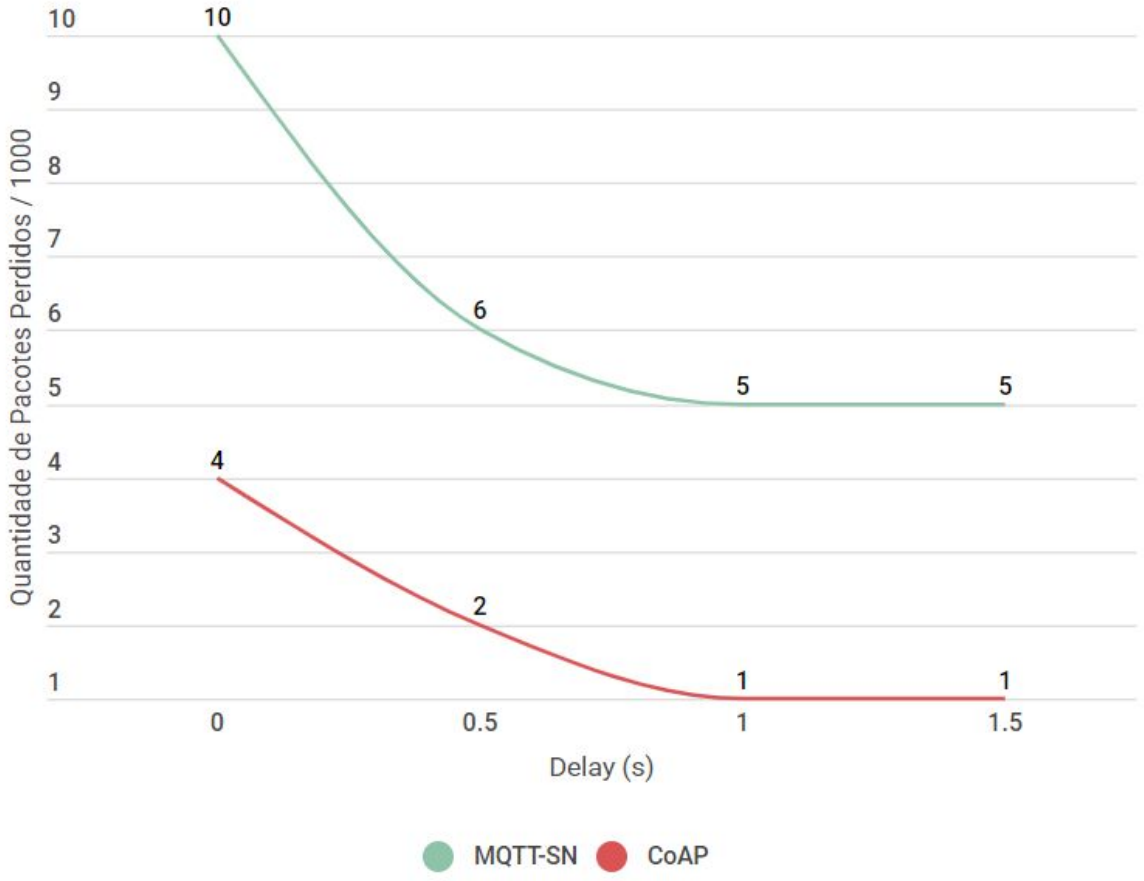


Gráfico 6 - Comparação de perda de pacotes com pacotes pequenos



Code pode ser visto em todas os gráficos anteriores, o tamanho da mensagem não é algo que influencie na latência, perda de pacotes e vazão. Um dos motivos para isso, é o poder de processamento do ESP8266, que consegue lidar bem com o máximo possível do tamanho do pacote.

6 CONCLUSÃO OU CONSIDERAÇÕES FINAIS

Ambos protocolos são ótimos para aplicações para Internet das Coisas. Ambos terão um ótimo desempenho em aplicações reais com sistemas com pouca capacidade operacional.

Mas vale ressaltar que o MQTT-SN tem um desempenho melhor no quesito velocidade, sendo uma opção melhor para aplicações que enviar um grande volume de dados à uma taxa de latência pequena, como o *Big Data.* Enquanto que o CoAP provou-se ser mais seguro e precisar menos retransmissões, já que perde menos pacotes, uma aplicação que precisa dessas características seria a transmissão de pacotes de voz.

Além dessa pesquisa, os mesmos testes podem ser aplicados à outros dois protocolos: o AMQP e o MQTT, onde ambos operam sob a camada TCP. Também seria interessante incluir o teste de consumo de energia, fazendo os cálculos em cima do consumo do rádio do ESP8266.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AGRAWAL, S., & VIEIRA, D. (2013). **A survey on Internet of Things** - DOI 10.5752/P.2316-9451.2013v1n2p78. *Abakós*, *1*(2), 78-95. https://doi.org/10.5752/P.2316-9451.2013v1n2p78.

AMARAN, M. H., *et al.* **A Comparison of Lightweight Communication Protocols in Robotic Applications.** Disponível em: https://reader.elsevier.com/reader/sd/pii/S1877050915038193?token=7CA6DC0B5ACC0C87A86FE83E6CDFA869C526525916C16E6B73BC8850741C20D00687BBD9B81DF1D8792D85A1FD1C39E8. Acesso em 04 de Novembro de 2019.

BEN-DAYA, M.; HASSINI, E.; BAHROUN, Z. **Internet of Things and supply chain management: a literature review.** Disponível em: https://www.tandfonline.com/doi/full/10.1080/00207543.2017.1402140. Acesso em 03 de Novembro de 2019.

BORMANN, C., CASTELLANI, A. P., SHELBY, Z., **CoAP: An Application Protocol for Billions of Tiny Internet Nodes.** Disponível em: https://ieeexplore.ieee.org/document/6159216. Acesso em 03 de Dezembro de 2019.

HUNKELER, U., TRUONG, H., L., STANFORD-CLARK, A. **MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks.** Disponível em:

https://sites.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/mqtt-s.pdf Acesso em 19 de Novembro de 2019.

MARTINS, I. R., ZEM, J. L., **ESTUDO DOS PROTOCOLOS DE COMUNICAÇÃO MQTT E COAP PARA APLICAÇÕES MACHINE-TO-MACHINE E INTERNET DAS COISAS.** Disponível em https://fatecbr.websiteseguro.com/revista/index.php/RTecFatecAM/article/view/41 Acesso em 04 de Novembro de 2019.

MAZZER, D., FRIGIERI, E. P., PEREIRA, L. F. P. P., **Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparação de Abordagens.** Disponível em: https://www.inatel.br/smartcampus/imgs/protocolos-para-iot-pt.pdf. Acesso em 03 de Novembro de 2019.

PRESS, Gil. **Internet of Things By The Numbers: Market Estimates And Forecasts.** Disponível em: https://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/#d637930b9194. Acesso em 01 de Novembro de 2019.

PORCIÚNCULA, C. B. da *et al.* **Constrained Application Protocol (CoAP) no Arduino UNO R3: Uma Análise Prática.** Disponível em: https://sol.sbc.org.br/index.php/wpietf/article/view/3212/3174. Acesso em 04 de Novembro de 2019.

RAMIREZ, J., PEDRAZA, C. **Performance analysis of communication protocols for Internet of Things platforms.** Disponível em https://ieeexplore.ieee.org/document/8088198. Acesso em 02 de Novembro de 2019.

SANTAELLA, L.; GALA, A.; POLICARPO, C.; GAZONI, R. **Desvelando a Internet das Coisas**. Revista GEMInIS, v. 4, n. 2, p. 19-32, 15 dez. 2013.

SANTOS, B. P. *et al.* **Internet das Coisas: da Teoria à Prática**. Disponível em: https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf Acesso em 02 de Novembro de 2019.

SHELBY, Z., HARTKE, K., BORMANN, C. **The Constrained Application Protocol (CoAP).** Disponível em: https://tools.ietf.org/html/rfc7252. Acesso em 20 de Outubro de 2019.

STANFORD-CLARK, A., TRUONG, H. L., **MQTT for Sensor Networks (MQTT-SN) Protocol Specification.** Disponível em: http://www.mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf. Acesso em 20 de Outubro de 2019.

TORRES, A. B. B., ROCHA, A. R., SOUZA, J. N. de.; **Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo.** Disponível em: http://wiki.stoa.usp.br/images/6/68/Mqt.pdf. Acesso em 04 de Novembro de 2019.

UPADHYAY, Y., BOROLE, A., DILEEPAN, D. **MQTT Based Secured Home Automation System.** Disponivel em https://ieeexplore.ieee.org/document/7570945. Acesso em 31 de Outubro de 2019.

XIA, Feng *et al*. **Internet of Things.** Disponível em : https://s3.amazonaws.com/academia.edu.documents/36946966/danainfo.acppwiszgmk2n0u279qu76contentserver.pdf?response-content-disposition=inline%3B%20filename%3DInternet_of_Things.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20191030%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20191030T223240Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=045ed1f1f3888716895c5c6d974bfa0e6e4a4d37b753edbee105a4209bb4aa81 Acesso em 30 de Outubro de 2019.