**COMPARAÇÃO ENTRE OS PROTOCOLOS DE COMUNICAÇÃO MQTT, COAP, MQTT-SN E AMQP**

Bianca de Espíndola

Lucas Rigotti Steglich

Patrick Davila Kochan

Randy Ramos Plácido

### RESUMO

Com o avanço da internet das coisas, surgem novas áreas no mundo tendo objetivo deste artigo é apresentar as comparações entre os protocolos de comunicação MQTT, CoAP, MQTT-SN e AMQP, e apresentar um teste de vazão de dados com o objetivo de realizar um *benchmarking* dos protocolos.

**Palavras-chave**: Protocolo, MQTT, CoAP, MQTT-SN, AMQP, mensagem, IoT.

### ABSTRACT

The objective of this academic article is to introduce a comparison between the communication protocols MQTT, CoAP, MQTT-SN and AMQP and present a throughput test to make a benchmarking of the protocols.

**Key words**: Protocol, MQTT, CoAP, MQTT-SN, AMQP, message, IoT.

### 1 **INTRODUÇÃO**

A Internet das Coisas conhecida como *IoT* (Internet of Things) surgiu com os avanços das diversas áreas tecnológicas, como na área de sistemas embarcados, principalmente na área de sensoriamento sem fio.

Como pode ser visto atualmente, o *IoT* tem recebido bastante atenção tanto na área acadêmica, quanto da área da indústria, devido ao seu grande potencial de uso nas mais diversas áreas. Sendo possível ver no nosso cotidiano, desde ao acordar ao olhar o relógio de pulso conectado ao celular, ao ir no shopping e ver quantas vagas tem disponível no estacionamento ao passar pela cancela.

A Internet das Coisas,se resume a uma extensão da Internet , que proporciona aos objetos do dia-a-dia, mas com capacidade computacional e de comunicação, se conectarem à Internet.

Com esse intuito, o objetivo do projeto e fazer uma comparação entre os protocolos de comunicação MQTT, COAP, MQTT-SN e AMQP.

### 2.1. **MQTT**

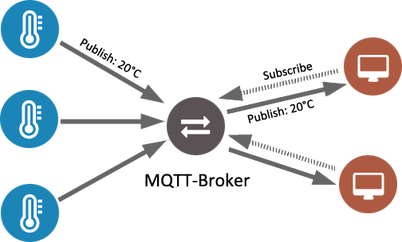
#### 2.1.1 HISTÓRIA

Desenvolvido em 1999 pela IBM, o protocolo MQTT (Message Queueing Telemetry Transport) é um protocolo aberto de comunicação. utiliza a o conceito de publicação e assinatura, trata-se de um protocolo de rede leve e flexível, pois é aplicável a redes com baixa largura de banda e alta latência. Esta aplicabilidade é ideal para um cenário de IOT em que existe uma quantidade de dispositivos estão conectados ao mesmo tempo de maneira pervasiva sendo capazes de interagir e cooperar entre si [Singh et al. 2014]. O protocolo MQTT é livre de *royalties* e foi padronizado pela OASIS.

Adotando um protocolo TCP para sua camada de transporte, junto do padrão de comunicação publicação e assinatura, todos os dados são enviados para a entidade central, chamada de *broker*, que é responsável por encaminhar as mensagens dos clientes que publicam os tópicos, até os clientes que tem a inscrição no mesmo tópico, desta forma a única entidade em que todos precisam se comunicar é o *broker* [MQTT Specification]. Por ter uma arquitetura leve e flexível, de fácil implementação em dispositivos e ambientes limitados, o MQTT acaba se tornando um dos melhores protocolos de rede para a IoT (YUAN, 2017), A Figura 01 demonstra uma aplicação simples do MQTT.

#### 2.1.2 MODELO DE PUBLICAÇÃO E ASSINATURA.

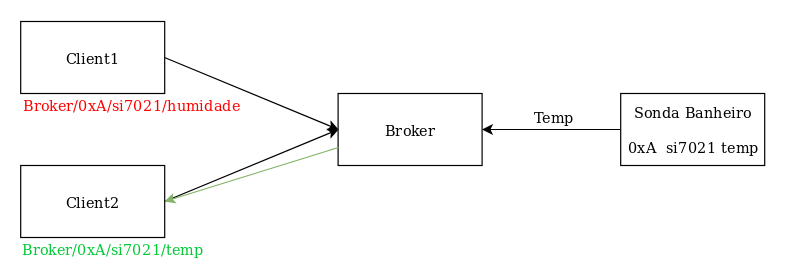
Figura 01: Arquitetura protocolo MQTT.

****

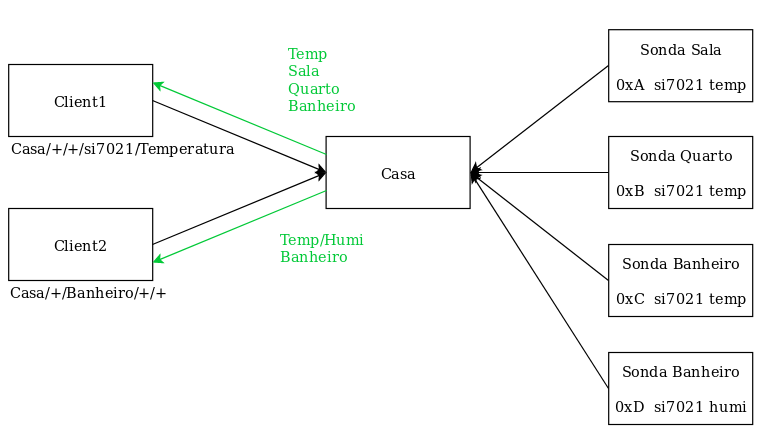
Fonte: https://pagefault.blog/2017/03/02/using-local-mqtt-broker-for-cloud-and-interprocess-communication/

Cliente é o dispositivo que comunica-se com o broker. Podendo se inscrever e publicar em qualquer tópico, cada tópico do MQTT é um endereço lógico formado por vários níveis, sendo cada um destes níveis separado por “/”, na Figura 02, é possível ver como o dado de uma sonda, com um certo endereço “0xA” de um dos seus sensores “si7021” envia um dado de temperatura, e apenas o cliente inscrito no tópico correto é capaz receberá a mensagem.

Figura 02:

****

Outro *token* de inscrição é o **+**, este comando quer dizer que qualquer tópico nesta posição que contém os endereços subsequentes deverão ser enviados, na Figura 03, temos 2 clientes, um deles está interessados em todas as temperaturas medidas pelos sensores si7021, de qualquer sensor em qualquer sonda, o outro cliente gostaria de saber qualquer dado de qualquer sensor que esteja apenas no banheiro, com estas inscrições o broker enviará as mensagens para os dispositivos corretos.

.

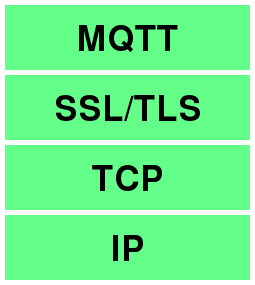
Outro caractere especial é o **#** que significa **qualquer** coisa abaixo do nível citado, por exemplo ao se inscrever e casa/#, no contexto citado anteriormente de uma casa, o cliente receberá mensagens de qualquer dispositivo, qualquer que seja o cômodo e de qualquer tipo de medição.

Este comando pode ser usado para receber todas as mensagens do broker, apenas quando assinado o tópico “#”. Estes caracteres especiais só podem ser usados para inscrição e nunca para a publicação de tópicos.

#### 2.1.3 **TOPOLOGIA MQTT**

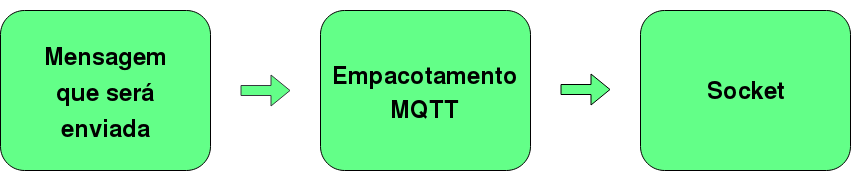
Segundo o modelo OSI, o MQTT encontra-se na camada de aplicação.

Figura 02: MQTT na camada de aplicação.

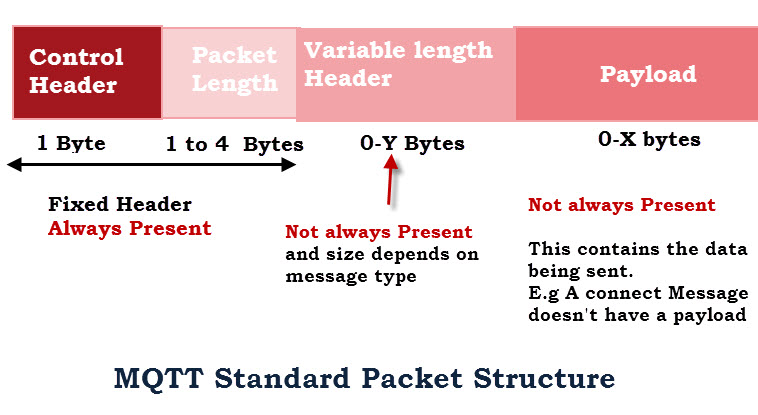


O cliente MQTT será responsável por encapsular o *payload*  que será enviado para o broker, o pacote montado pode-se usar SSL/TLS para adicionar segurança.

Figura 03: Caminho Encapsulamento do payload.



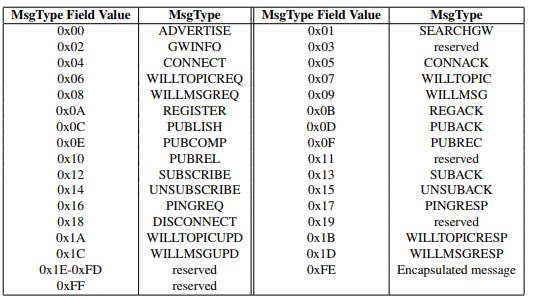
O processo de encode do pacote mqtt é relativamente simples, sendo possível de ser realizado em dispositivos embarcados, mesmo utilizando SSL/TLS, a maioria dos dispositivos embarcados modernos já possui poder de processamento capaz de enviar sem problemas pacotes MQTT utilizando SSL/TLS (RAVI KODALLI, 2017).

Figura 04: Estrutura Padrão do Pacote.

Fonte: <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/>.

**Cabeçalho de controle:** Neste byte inicial há os valores de controle que o MQTT implementa como “CONNECT” “SUBSCRIBE” “PUBLISH” ele diz do que se trata o pacote, podendo ser também um ACK;

Tabela 01: Estrutura Padrão do Pacote.



Fonte: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

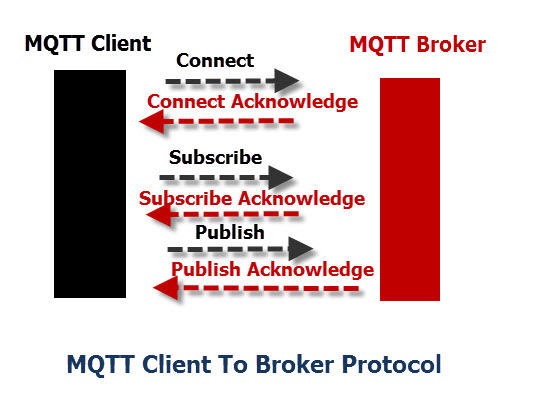
**Tamanho do pacote:** Neste campo apenas dirá qual o tamanho do dado puro contido na mensagem, porém no campo **cabeçalho** em que pode ter-se os mais variados tipos de informações, com base no tipo de mensagem do cabeçalho de controle, esta parte da mensagem pode não existir, ou como pode haver mais 3 sub-campos preenchidos.

***Payload***, trata-se dos dados em si que foram enviados, caso seja um campo de publish por exemplo, este campo existe, mas se o cabeçalho de controle disser que se trata apenas de um ACK, então o campo payload não estará contigo no pacote. Neste processo de troca de mensagem, existe algumas terminologias específicas:

* **CONNECT**: Inscrição de um cliente a um broker;
* **PUBLISHED:** Envio de mensagens;
* **SUBSCRIBE:** Inscrição em um tópico para receber mensagens.

O processo e conexão **CONNECT** é feito através de uma transação TCP/IP, o cliente recebe o pacote CONNACK para verificar que a conexão ocorreu com sucesso, a partir desta conexão feita, o cliente está apto a se inscrever e publicar em tópicos. Com esta conexão estabelecida o cliente pode solicitar ao broker os comandos PUBLISH e SUBSCRIBE, o broker irá responder com ACK as transações através das respostas PUBACK e SUBACK. (SHUBHANGI SHINDE, 2016).

Figura 05: Estrutura Padrão do Pacote.

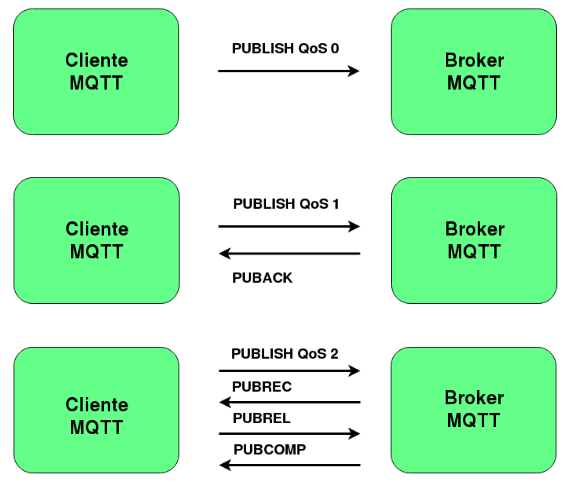


Referencia : http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/

#### 2.1.4 QUALIDADE DO SERVIÇO

O MQTT introduz 3 níveis de qualidade de serviço, este está atrelado em quão garantida é a entrega de um pacote.

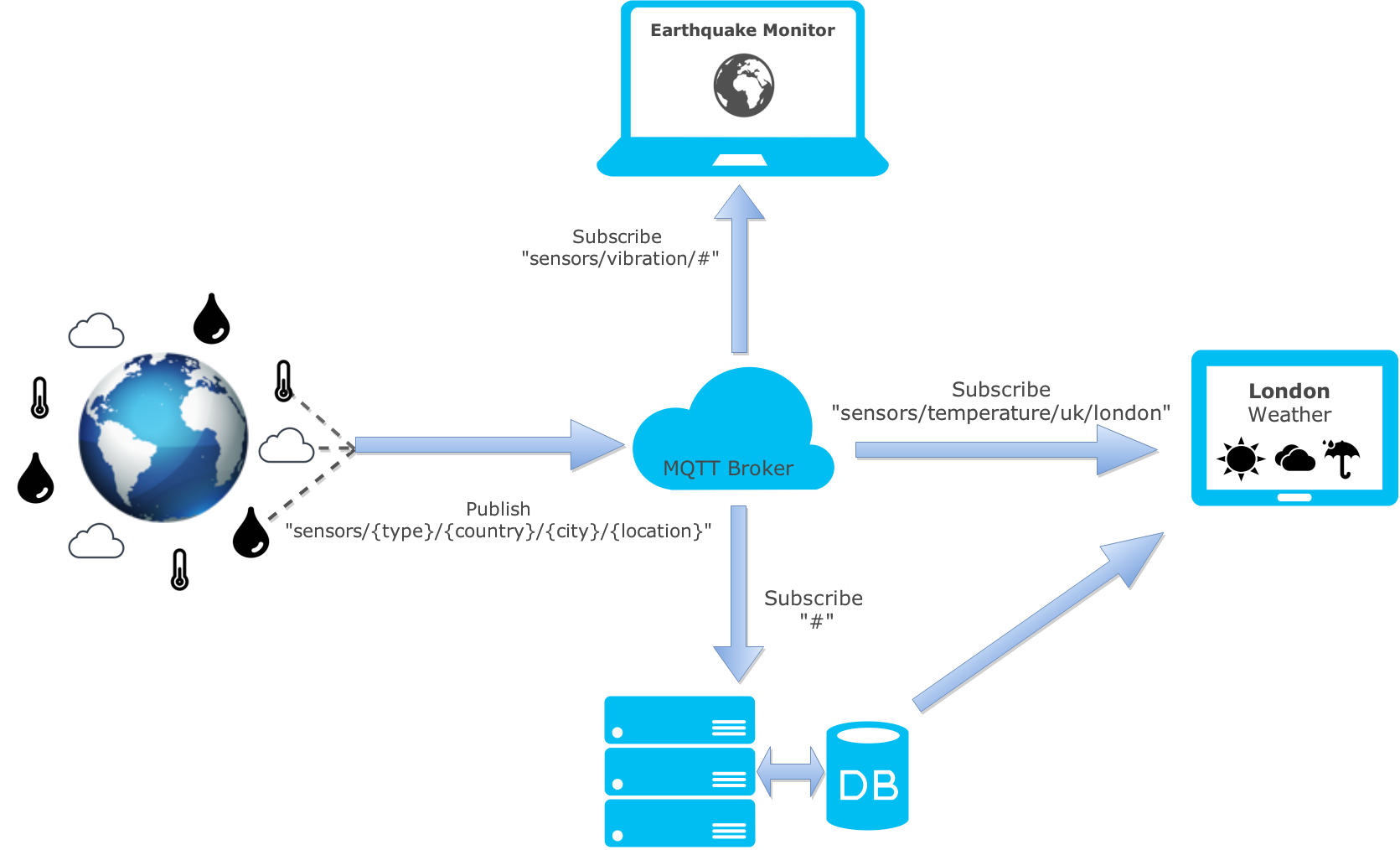
Figura 06: Estrutura de envio do Pacote.



De maneira geral, o nível de QOS está relacionado a quantos ACKS o cliente deseja receber para confirmar que o dado foi entregue, QoS 1 já é o suficiente para a maioria das aplicações porém mensagens importantes de configuração ou informações críticas pode ser necessário enviar com uma garantia maior, assim como mensagens menos importantes podem ser colocadas sem garantia de entrega, essa é uma das flexibilidades do MQTT em seu controle de mensagens.

#### 2.1.5 ARQUITETURA DA REDE

Figura 07: Arquitetura da Rede.



Fonte: .https://zoetrope.io/tech-blog/brief-practical-introduction-mqtt-protocol-and-its-application-iot/

A implementação comum do MQTT, se trata de um broker conectado a vários clientes, alguns deles apenas estão adquirindo todos os seus dados como o banco de dados demonstrado inscrito no tópico “#”, outros são estações ao redor do mundo, que publicam a temperatura de um local de tempos em tempos, e outros clientes, apenas adquirem o dado que lhes interessa como a previsão do tempo ou a temperatura atual.

### 2.2. MQTT-SN

O MQTT-SN é considerado uma versão adaptada do MQTT, com às peculiaridades de um ambiente de comunicação sem fio. Os links de rádio sem fio,tem em geral taxas de falha mais altas do que as com fio, devido à suscetibilidade ao desbotamento e perturbações de interferência.Eles também têm uma taxa de transmissão menor. (STANFORD-CLARCK,2013)

Além disso, para não perderem seus pacotes durante as transmissões, seus pacotes têm um comprimento muito curto. No caso do IEEE 802.15.4, o comprimento do pacote na camada física é limitado a 128 bytes. Metade desses 128 bytes pode ser tomada afastado pelas informações gerais exigidas pelas funções de suporte, como camada MAC, rede, segurança, etc.

O MQTT-SN também é otimizado para implementação em dispositivos de baixo custo, operados por bateria, com processamento limitado e recursos de armazenamento. O MQTT-SN foi projetado de forma que seja independente dos serviços de rede subjacentes. Qualquer rede que fornece um serviço de transferência de dados bidirecional entre qualquer nó e um em particular (um gateway) ser capaz de suportar MQTT-SN. (STANFORD-CLARCK, 2013)

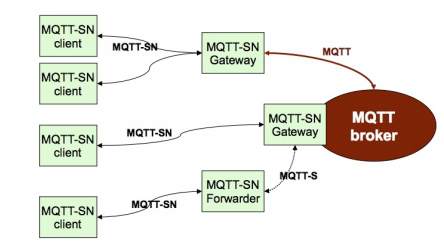
#### 2.2.1. **ARQUITETURA DO MQTT-SN**

A arquitetura do MQTT-SN vista na Figura 8. Ela é composta por três componentes: clientes MQTT-SN, gateways MQTT-SN e forwarders (encaminhadores) MQTT-SN. Clientes MQTT-SN se conectam a um broker MQTT por meio de um gateway MQTT-SN, utilizando o protocolo MQTT-SN. Um gateway MQTT-SN pode ser capaz ou não a se agregar a um broker MQTT. No caso de um gateway independente, o protocolo MQTT é usado entre o broker e o gateway. A sua função principal é a tradução entre os protocolos MQTT-SN e o MQTT.(ROCHA,2018)

Os clientes podem se conectar ao *gateway* através do *forwarder* caso não estejam conectados a uma rede. O *forwarder* simplesmente vai encapsular os pacotes MQTT-SN que recebe e os encaminha sem nenhuma alteração para o *gateway*.

Na outra direção do *forwarder* ele vai desencapsular os pacotes recebidos do *gateway* e os envia para os clientes sem alterar os pacotes. Dependendo de como um gateway realiza a conversão de protocolo entre MQTT e MQTT-SN, podemos diferenciar entre dois tipos de gateway, ou seja, gateway transparentes e agregados.

Figura 08: arquitetura MQTT-SN.

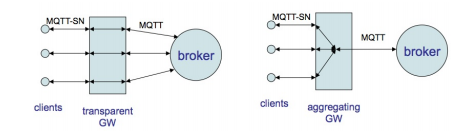
****

Fonte:(STANFORD-CLARCK,2013).

Para cada cliente MQTT-SN conectado, um *gateway* transparente configurará e manterá uma conexão MQTT-SN com o broker MQTT. Essa conexão MQTT é reservada exclusivamente para a troca de mensagens de ponta a ponta e quase transparente entre cliente e servidor. Haverá tantas conexões MQTT entre o gateway e o servidor como clientes MQTT-SN conectados. O *gateways* transparente executará uma tradução de “sintaxe” entre os dois protocolos. Como todas as trocas de mensagens são de ponta a ponta entre o cliente MQTT-SN e o Servidor MQTT, todas as funções e recursos implementados pelo servidor podem ser oferecidos ao cliente.(STANFORD-CLARCK,2013)

Embora a implementação do *gateway* transparente seja mais simples quando comparada com a de um *gateway* agregador, requer que o servidor MQTT suporte uma conexão separada para cada cliente ativo. Algum servidor MQTT implementações podem impor uma limitação no número de conexões simultâneas que elas suportam.

Figura 09: gateway.



Fonte:(STANFORD-CLARCK,2013).

Em vez de ter uma conexão MQTT para cada cliente conectado, um gateway de agregação terá apenas um MQTT conectado com o servidor.

O *gateway* então decide quais informações serão dadas ao servidor. Embora sua implementação é mais complexo do que o de um *gateway* transparente, um *gateway* agregado pode ser útil no caso de *WSNs* com número muito grande de *SAs* porque reduz o número de conexões MQTT que o servidor deve suportar ao mesmo tempo.

As mensagens são enviadas de um publicador para um tópico no broker através do *gateway*. O *gateway* transfere as mensagens para os assinantes para este tópico no broker. Para transferir as mensagens dos dois protocolos, o MQTT e o MQTT-SN começam com a mensagem *CONNECT* do cliente para o broker. As respostas do servidor com uma mensagem *CONNACK*. Para finalizar a comunicação, o cliente envia uma mensagem *DISCONNECT* para o servidor.

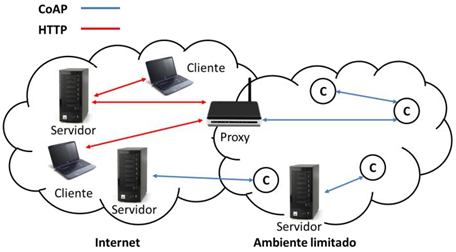
### 2.3. **COAP: *CONSTRAINED APPLICATION PROTOCOL***

Proposto em 2014 pelo grupo CoRE (*Constrained RESTful Environment*) da IETF (*Internet Engineering Task Force)* e definido pela RFC (*Request for Comments*) 7252, o CoAP é um protocolo muito novo que faz uso dos protocolos UDP e IP para realizar comunicação M2M (*machine to machine*). (IETF, 2014)

O CoAP foi projetado para ser um protocolo focado em *Internet of Things* (IoT) leve e econômico, utilizando recursos mínimos tanto no dispositivo quando na rede, sendo mais eficiente energeticamente que os demais protocolos de comunicação e uma excelente alternativa para dispositivos com baixa capacidade de memória. Suas mensagens são pequenas, causando pouca fragmentação na camada de enlace. (MAZZER; FRIGIERI; PARREIRA, 2015)

Sua construção foi modelada de maneira a facilitar sua integração com o protocolo HTTP, utilizando métodos de solicitação e resposta baseados no modelo REST (*Representational State Transfer*): servidores disponibilizam recursos em um URL (*Uniform Resource Locator*) e os clientes acessam esses recursos através de métodos GET, PUT, POST e DELETE. Como ambos os protocolos compartilham este modelo, pode-se facilmente conectar esses dois protocolos utilizando um servidor *proxy*. (MAZZER; FRIGIERI; PARREIRA, 2015).

Figura 10: exemplo de ambiente CoAP interligado a ambiente HTTP

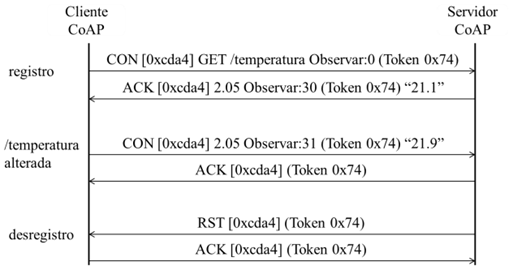


(MAZZER; FRIGIERI; PARREIRA, 2015)

Com forte embasamento no protocolo HTTP, o CoAP implementa funções interessantes que não são implementadas no primeiro, como a função de observação e descoberta. Para resolver o problema da observação, o CoAP utiliza a *flag observe*: ao enviar uma mensagem GET de um dispositivo x com uma *flag observe* para um dispositivo y, este irá verificar periodicamente se uma mudança de estado ocorreu na variável, enviando uma notificação para o observador se a mudança aconteceu. A função de descoberta identifica novos dispositivos na rede, fazendo com que os servidores armazenam uma lista de dispositivos que estão em sua vizinhança e compartilhem na rede informações sobre os recursos desses dispositivos e os tipos de mídia que eles suportam. (BORMANN, 2016)

Em termos de Qualidade de Serviço (Quality of Service ou simplesmente QoS), este protocolo apresenta dois tipos de mensagem. A mensagem de confirmação, onde uma mensagem ACK (acknowledge) confirma ao solicitante que a mensagem foi recebida e a mensagem não confirmável, para casos onde a informação de que a mensagem foi recebida ou não pelo outro dispositivo não é interessante. Para este segundo tipo, uma nova mensagem não confirmável é repassada como resposta ao cliente. Uma mensagem RST (reset) é retornada pelo servidor se ele não for capaz de fornecer uma resposta. (CASSINELLI; MIRANDA; VALE, 2018)

Figura 11: exemplo de utilização do recurso de observação e de troca de mensagens CoAP.



(MAZZER; FRIGIERI; PARREIRA, 2015)

Como mecanismo de segurança o CoAP implementa o DLTS (Datagram Transport Layer Security) para garantir a troca segura de mensagens na camada de transporte, prevenindo ataques de escuta, adulteração e da forja de dados. (CASSINELLI; MIRANDA; VALE, 2018)

Um bom exemplo de utilização do CoAP, seria sua implementação em um *hardware* que precisa de um tratamento de entrada de energia muito baixa, com uma bateria pra alimentá-lo. Este protocolo vem sendo largamente utilizado nas áreas de energia inteligente, automação industrial e predial. (CASSINELLI; MIRANDA; VALE, 2018)

### 2.4. **AMQP**

O AMQP, *Advanced Message Queuing Protocol*, é um protocolo aberto de comunicação em rede, bem como o *HTT*P ele permite a troca de dados para comunicação entre diferentes aplicações. Dentro deste protocolo há filas, *exchanges* e *Bindings*, os quais são chamados de entidades e serão abordados a seguir.

É utilizado de um *Broker*, intermediário entre o comunicador da mensagem e os seus receptores. Como já dito, o *Broker* receberá a mensagem das aplicações que as publicam e irá direcionar às aplicações que processam mensagens.

O processo então é realizado da seguinte maneira: as aplicações que publicam mensagens fazem as publicações que são direcionadas às *exchanges*. As *exchanges* irão distribuir as mensagens, cópias das mesmas quando necessário, para filas que utilizam *bindings*, regras e restrições para dar prioridade a receptores desejados.

Após esse passo, os *Brokers* comunicam através de mensagens os consumidores que acessam ou assinam as filas. Os próprios consumidores podem buscar e processar as mensagens da fila, dada a demanda e prioridades já estabelecidas.

Quando uma mensagem é publicada, os produtores podem classificar diversos atributos à ela, comumente conhecidos como *metadados da mensagem*. Caso seja de importância, o *Broker* pode vir a utilizar alguns destes atributos, sendo reservados aqueles de utilização restrita, utilizado em outros momentos pelas aplicações que recebem mensagens.

Para que o fator segurança não falhe com facilidade, critério de importância se tratando de troca de mensagens, na estrutura do AMQPhá a implementação de *acknowledgments* *de mensagens*. Sua função é fazer com que o consumidor notifique o *Broker* do envio da mensagem. Esta notificação pode ser feita automaticamente ou posteriormente.

Durante a utilização de *acknowledgments*, um *Broker* removerá totalmente a mensagem da fila somente quando a mesma tiver sido notificada que recebeu uma mensagem, seja uma ou múltiplas mensagens.

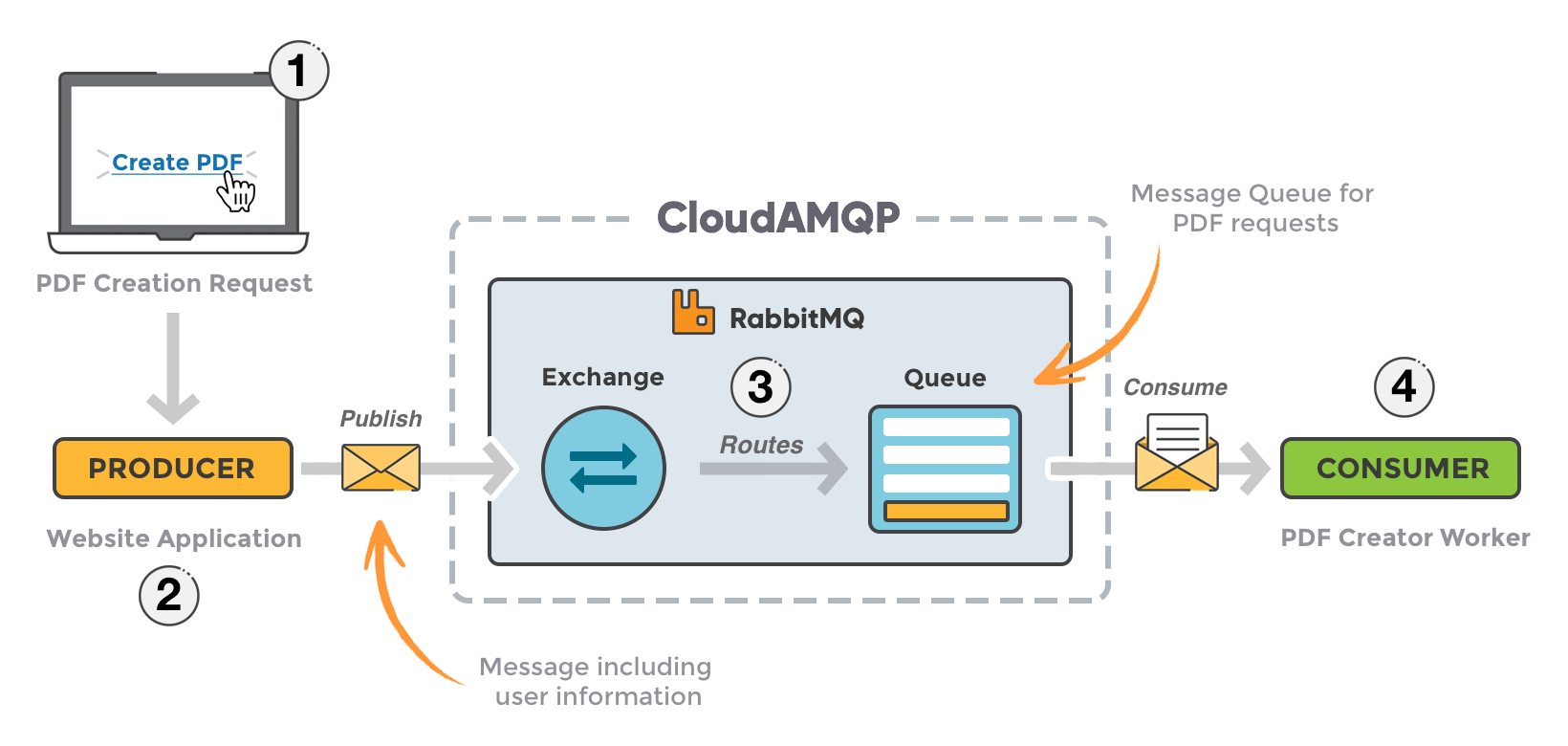
Podem haver situações onde uma mensagem não pode ser direcionada pelo *Broker* ou pelas aplicações de publicação de mensagens. Nestes casos, esta mensagem pode ser retornada para as aplicações que a publicaram, descartada ou manipulada. A manipulação desta mensagem pode ser realizada caso seja implementada uma extensão chamada de *dead letter queue*, onde os publicadores da mensagem podem escolher qual será o destino destes dados, já proporcionados alguns parâmetros.

#### 2.4.1. **ARQUITETURA DO AMQP E RABBITMQ**

Muito comumente associado ao AMQP é a utilização do software *RabbitMQ*, que gera o processo a partir do protocolo.

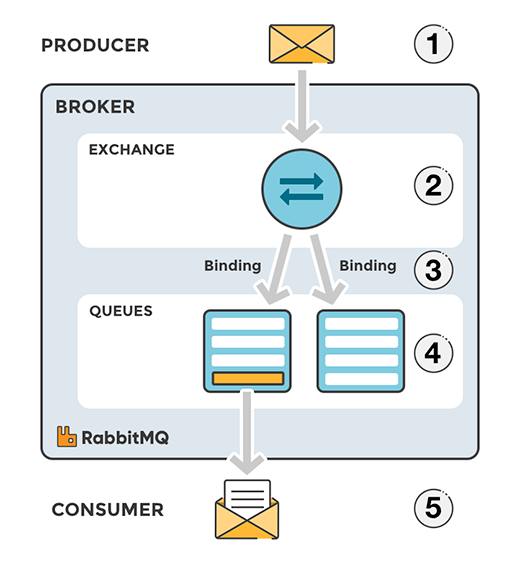
Na Fig. 12 é apresentado o fluxo das funções realizadas para a criação, publicação e envio, manejo e distribuição, entrega e recebimento das mensagens.

Figura 12: fluxo de atividades.



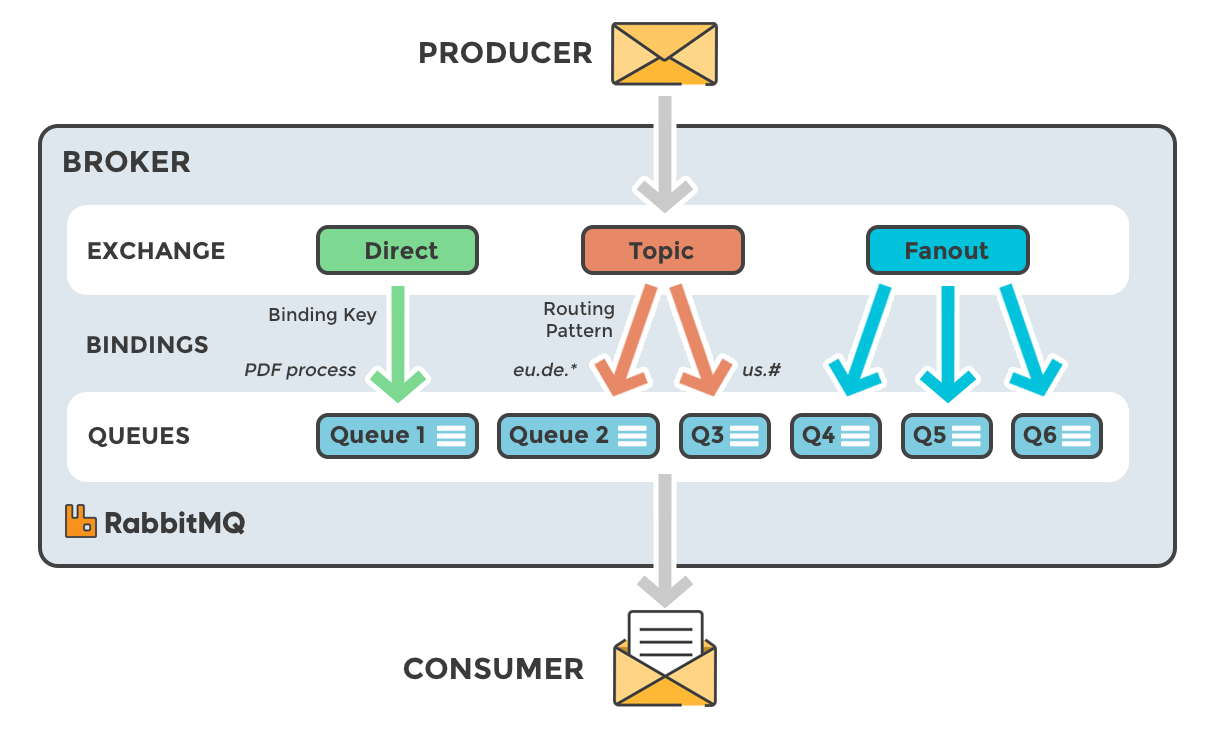
Na Fig. 13 pode-se observar a interação do produtor da mensagem e os demais agentes, como o *Broker*, *Exchanges*, *Bindings*, filas e o consumidor.

Figura 13: *exchanges.*



Na Fig. 14 vemos a interação entre os agentes do protocolo, onde é destacada a interação entre as *Exchanges* e as filas.

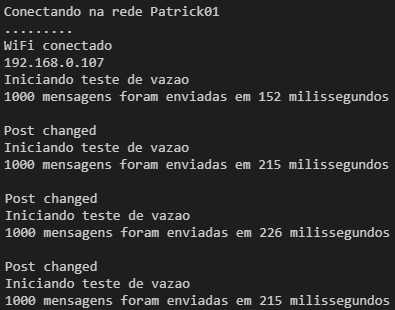
Figura 14: filas e prioridades



### 2.5. COMPARAÇÃO MQTT E COAP

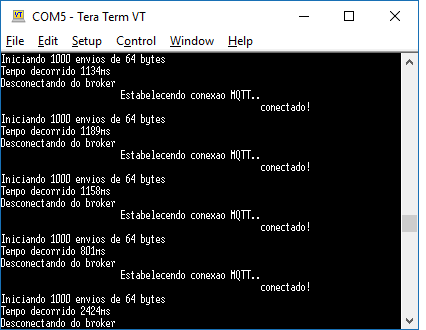
Um teste de vazão foi realizado para esses dois protocolos, o MQTT e o CoAP. Esse teste consistia no envio de 1000 (mil) mensagens, registrando o momento em que ela começaram a ser transmitidas e o momento em que a transmissão foi encerrada. Então o tempo de envio dessas mensagens foi obtido.

Figura 15: Teste de vazão de mensagens utilizando CoAP.



Para o CoAP, as mil mensagens foram enviadas em uma média de 202 milissegundos, já para o MQTT, uma média de 1341 milissegundos foi obtido como resultado. Assim podemos concluir que o protocolo COAP apresenta maior vazão em relação ao protocolo MQTT.

Figura 16: Teste de vazão de mensagens utilizando MQTT..



### 2.6. CONSIDERAÇÕES FINAIS

Os protocolos utilizados em aplicações de internet das coisas ou IoT (*Internet of Things*) vem sendo explorados cada vez mais no mercado. Essa tendência é evidenciada quando observamos a miniaturização de componentes eletrônicos disponibilizados no mercado.

O MQTT, CoAP, MQTT-SN e o AMQP são apenas uma pequena amostra da imensa gama de protocolos focados em IoT e cada um possui suas peculiaridades. Em especial, o MQTT possui bastante presença nas aplicações e é amplamente utilizado. O *Constrained Application Protocol* também demonstra sua expressividade em aplicações, com sua proposta de garantir suavidade na curva de transição do HTTP para esse novo mercado.

### REFERÊNCIAS

BORMANN, Carsten. RFC 7252 Constrained Application Protocol: Specification. 2016. Disponível em: <http://coap.technology/>. Acesso em: 08 maio 2019.

CASSINELLI, Fernanda; MIRANDA, Lucas; VALE, Lucas do. Constrained Application Protocol. 2018. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/coap/coap.html>. Acesso em: 08 maio 2019.

CLARK, Andy Stanford; TRUONG, Hong Linh. MQTT For Sensor Networks

Stanford-Clark, A., & Truong, H. L. (2013). MQTT for sensor networks ( MQTT-SN) protocol specification. IBM.

Kodali, R. K., & Soratkal, S. R. (2017). MQTT based home automation system using ESP8266. In IEEE Region 10 Humanitarian Technology Conference 2016, R10-HTC 2016 - Proceedings. <https://doi.org/10.1109/R10-HTC.2016.7906845>

Yuan, M. (2017). Conhecendo o MQTT. IBM.

OASIS. (2014). MQTT Version 3.1.1. OASIS Standard.

Shinde, S. A., Nimkar, P. A., Singh, S. P., Salpe, V. D., & Jadhav, Y. R. (2016). MQTT - Message queuing telemetry transport. International Journal of Research.

INTERNET ENGINEERING TASK FORCE (Estados Unidos). The Constrained Application Protocol (CoAP). 2014. Disponível em: <https://tools.ietf.org/html/rfc7252>. Acesso em: 12 abril 2019.

MAZZER, Daniel; FRIGIERI, Edielson P.; PARREIRA, Luís Felipe C. G. Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparação de Abordagens.

ROCHA, Marcelo Marques da. **DESENVOLVIMENTO OPEN-SOURCE PARA A INTERNET DAS COISAS (ARQUITETURAS PARA INTERFACES WEB E MÓVEL).** 2018. 148 f. TCC (Graduação) - Curso de Tecnologia em Sistemas de Computação, Universidade Federal Fluminense, Niteroi, 2018. Disponível em: <http://riut.utfpr.edu.br/jspui/bitstream/1/3746/3/PG\_PPGCC\_M\_Rocha%2c%20Helbert%20da\_2018.pdf>. Acesso em: 24 jun. 2019.