Uma Análise dos Protocolos de Comunicação para Internet das Coisas*

Arthur Brito Cosmi[1], **Vinícius Fernandes Soares Mota**[1]

[1]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Vitória – ES – Brasil

arthurcosmi@gmail.com, vinicius.mota@inf.ufes.br

Abstract. *With the growth of devices connected to the Internet, several application layer protocols specific for the Internet of Things (IoT) appear every day. This high number of protocols bring doubts to developers about what to use in their projects. This work aims to assist developers in the decision making in their IoT projects. This work presents a qualitative analysis of the MQTT, MQTT-SN, CoAP and AMQP protocols. In a qualitative way, the domains of application and requirements of the applications were analyzed to map the protocol choice. After that, the MQTT and CoAP protocols were analyzed quantitatively using devices with low processing and memory capabilities. Results show that MQTT outperforms CoAP when there are several other devices accessing the same access point.*

Resumo. *Com o crescimento de dispositivos conectados à Internet, diversos protocolos da camada de aplicação específicos para Internet das Coisas (IoT) surgem a cada dia. Esta alta quantidade de protocolos trazem dúvidas aos desenvolvedores sobre o que utilizar em seus projetos. O objetivo deste trabalho é auxiliar os desenvolvedores na tomada de decisão em seus projetos IoT. Este trabalho apresenta uma análise qualitativa dos protocolos MQTT, MQTT-SN, CoAP e AMQP. De modo qualitativo, foram analisados os domínios e requisitos das aplicações para mapear a escolha dos protocolos. Após isso, os protocolos MQTT e CoAP foram analisados quantitativamente utilizando-se dispositivos com baixo poder de processamento e memória. Os resultados demonstram que o MQTT tem melhor desempenho que o CoAP quando existe diversos outros dispositivos acessando o mesmo ponto de acesso.*

1. Introdução

A Internet das Coisas (*Internet of Things* - IoT) é uma área tecnológica em constante crescimento nos últimos anos. De fato, são estimados 50 bilhões de objetos inteligentes e conectados à Internet em 2020, propiciando a cada dia dezenas de soluções para melhorar a qualidade de vida das pessoas e automatizar tarefas em residências, empresas e indústrias [Atzori et al. 2010]. Além disso, nos cenários urbanos, os objetos conectados trazem uma nova perspectiva para interação entre residentes e espaços físicos de uma cidade [Zanella et al. 2014].

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), do CNPq, da FAPES e do projeto RNP FUTEBOL.

No contexto da computação urbana, o fluxo de dados consiste em sensoriar, transmitir e armazenar para, posteriormente, analisar os dados coletados. Sistemas embarcados, conhecidos como *System-On-Chip* (SOC), permitem a utilização de sensores, atuadores e transmissão de dados em *hardwares* com tamanho restrito. No geral, isto implica em processamento, memória primária e armazenamento limitados. Por outro lado, estes dispositivos apresentam baixo consumo energético.

No entanto, projetos de sensoriamento e transmissão de dados requerem que os desenvolvedores definam a arquitetura do hardware e os protocolos de comunicação que serão utilizados. O projeto de um dispositivo leva em consideração, custo, capacidade de processamento, memória e armazenamento, rede de comunicação e fonte de energia. Já a escolha de protocolos de comunicação dependerá do desempenho da arquitetura bem como de características da aplicação.

Diversos protocolos de comunicação, que atuam na camada de aplicação, específicos para Internet das Coisas têm sido propostos na literatura [Al-Fuqaha et al. 2015]. Os protocolos da camada de aplicação IoT permitem uma melhor interação entre os dados sensoreados pelos dispositivos e à aplicação alvo. Esses protocolos vão de versões similares ao HTTP modificada para o mundo de objetos, como o CoAP, à protocolos de distribuição de mensagens em modo *multicast*, como o MQTT [Naik 2017]. O grande problema é que não existe uma definição muito precisa de quando se deve utilizar determinado protocolo, nem quais os impactos provocados por essa escolha. A partir da análise das aplicações disponíveis no mercado é possível mapear os domínios em que esses protocolos podem ser utilizados de maneira mais eficiente e analisar as métricas de desempenho dos mesmos.

Com o objetivo de auxiliar desenvolvedores na escolha de protocolos de aplicação em projetos que utilizem SOCs, este trabalho apresenta um estudo qualitativo e quantitativo dos principais protocolos. A partir de uma ampla revisão da literatura, apresentamos um estudo quantitativo mapeando como cada protocolo analisado reage a um conjunto de requisitos de aplicações IoT. Após isso, dois protocolos foram implementados para serem analisados e comparados em um dispositivo SOC com processamento e memória limitado, o ESP8266. O *testbed* desenvolvido permitiu analisar e comparar o desempenho dos protocolos quando o dispositivo estava conectado isoladamente à rede sem fio e quando havia tráfego de fundo. Foram analisados a taxa de entrega, o tempo de transmissão e a retransmissão de pacotes. As principais contribuições deste trabalho são:

- Classificar e categorizar, qualitativamente, os protocolos de aplicação de acordo com os requisitos de cada domínio de aplicação. Foram considerados os domínios residenciais, comerciais e industriais;
- Prototipar uma aplicação e analisar, quantitativamente, o desempenho de protocolos de aplicação em relação à taxa de entrega, retransmissão de pacotes e tempo de transmissão, considerando um cenário isolado e um cenário com tráfego de fundo;

O restante deste trabalho está organizado da seguinte forma: A Seção 2 faz uma revisão dos protocolos de aplicação IoT. Os trabalhos relacionados são apresentados na Seção 3. Uma avaliação qualitativa dos protocolos de aplicação é discutida na Seção 4. A Seção 5 apresenta uma comparação quantitativa, utilizando um *testbed*, entre os protocolos MQTT e CoAP. Os resultados dos experimentos são discutidos na Seção 6. Por fim, a Seção 7 conclui este trabalho e apresenta as considerações finais.

2. Protocolos da camada de aplicação

Esta seção introduz os principais protocolos utilizado para transmissão de dados de dispositivos móveis para a base de dados das aplicações em Internet das coisas. Os protocolos que conseguiram aderência da indústria, tais como *Microsoft*, *Amazon*, *Google*, entre outras, são os protocolos MQTT (*Message Queuing Telemetry Transport*), MQTT-SN, CoAP (*Constrained Application Protocol*), AMQP (*Advanced Message Queuing Protocol*). Deste modo, esses protocolos serão descritos brevemente na Seção 4 para apresentarmos uma avaliação qualitativa sobre os mesmos.

O *Message Queuing Telemetry Transport* - MQTT - é um protocolo de comunicação sobre a pilha TCP/IP, criado pela IBM em 1999. O MQTT ganhou popularidade por utilizar o modelo *Publisher/Subscriber* [Mazzer et al. 2015]. Neste modelo, toda comunicação é intermediada por um dispositivo chamado concentrador (*Broker*). Dispositivos chamados *Publishers* enviam suas mensagens para um *Broker* utilizando uma estrutura chamada de *tópico*, que funciona como um assunto de interesse. Dispositivos chamados de *Subscribers* podem se inscrever em vários tópicos junto ao *Broker*. O *Broker* encaminha as mensagens publicadas em um determinado tópico para todos os dispositivos inscritos no mesmo.

O MQTT-SN foi desenvolvido para permitir o uso do MQTT em redes não TCP/IP, tais como *Zigbee* ou UDP/IP. Para isto, o MQTT-SN implementa seu próprio controle de fluxo e retransmissão, além de possuir algumas características interessantes como a criação de um mecanismo de identificadores de tópicos, onde cada cliente pode pedir ao *Broker* que gere um número para o tópico. Com isso, ao invés de utilizar o nome do tópico em UTF-8 para a troca de mensagens, é utilizado o identificador [Truong and Linh 2013].

Message-Oriented Middleware ou MOM é uma arquitetura que funciona como uma camada única de compartilhamento de informações para conectar sistemas distribuídos. O *Advanced Message Queuing Protocol* - AMQP - veio para padronizar essa comunicação entre MOM's de forma a garantir a interoperabilidade e confiabilidade na entrega das mensagens [Appel et al. 2010]. Além disso, este protocolo define como as mensagens serão publicadas e consumidas e o tempo de vida delas. O AMQP tem a vantagem de manter as mensagens no servidor e enviá-las para o consumidor assim que este estiver disponível. Assim, vários dispositivos podem utilizar funções de *sleep* e economizar energia o que é bastante aceitável em aplicações IoT. Outra vantagem é a segurança em cima das mensagens enviadas. A característica de mensagens imutáveis garante o envio das mensagens de ponto a ponto.

O *Constrained Application Protocol* - CoAP - é um protocolo desenvolvido para ser compatível com o modelo HTTP REST. O CoAP utiliza comunicação no modelo requisição/resposta. Neste modelo, dispositivos clientes requisitam uma informação (GET) e os dipositivos, atuando como servidores, respondem a informação (POST) [Mazzer et al. 2015]. O protocolo CoAP foi projetado com um cabeçalho fixo de quatro *bytes* e utiliza o UDP como protocolo de transporte. As requisições são realizadas por meio de URI's. Além disso, o CoAP implementa sua própria forma de confiabilidade para entrega de mensagens e um mecanismo de retransmissão. Para diminuir ainda mais o tráfego de rede, o protocolo permite o envio de mensagens em *piggyback* utilizando *MessageIDs* e *Tokens* para garantir o envio correto da informação [Shelby et al. 2014].

3. Trabalhos Relacionados

Com o aumento das aplicações na área de IoT, vários estudos foram realizados para verificar a eficiência dos protocolos [Mun et al. 2016, Chaudhary et al. 2017]. A heterogeneidade dos dispositivos bem como dos meios físicos de comunicação, tais como redes *wireless*, cabeada e de sinais 2/3/4G, afetam o desempenho dos protocolos. Portanto, a escolha do protocolo de comunicação torna-se um desafio aos desenvolvedores de aplicações IoT.

Em [Mun et al. 2016], os autores compararam cinco protocolos de comunicação (MQTT, CoAP, MQTT-SN, WEBSOCKET, TCP). Para isto, os autores montaram um *testbed* utilizando dispositivos *Raspberry Pi* comunicando-se com uma base de dados local e outra em Tóquio. Foram medidos o tempo de transmissão, consumo total de energia e uso de memória e CPU. Nos cenários analisados, o desempenho do TCP e do *WebSocket* foram superiores. Para pacotes menores que 1024 *bytes*, o CoAP foi melhor que o MQTT. No entanto, o CoAP teve o pior desempenho em memória utilizada quando houveram retransmissões de pacotes. Como conclusão, para que o CoAP seja utilizado em dispositivos restritos, o tamanho do pacote deve ser menor que 1 KB. O MQTT, para o mesmo fim, deve ter até 1,5 KB e o MQTT-SN não está preparado para a aplicação em sistemas externos.

Em [Chaudhary et al. 2017], os autores compararam o desempenho do MQTT,CoAP e AMQP para o envio constante de dados em redes cabeadas, *WiFi* e 2/3/4G. Os autores mensuraram o *Overhead* e a vazão das mensagens. O *testbed* era composto por um servidor de alto desempenho sendo utilizado como *broker* do MQTT e do AMQP e como servidor do CoAP e um *Raspberry Pi-3*, utilizado como cliente MQTT, AMQP e CoAP. Além disto, um computador portátil foi utilizado como cliente. Os testes realizados faziam o envio sucessivo, com intervalos pequenos entre envios, de mensagens com tamanhos diferentes com base 10 (10,100,1000,100000). Os autores concluíram que em aplicações com alta taxa de mensagens, o MQTT QoS1 e 2 utiliza muita largura de banda por causa da retransmissão de pacotes. Já o CoAP consome muito pouco recurso dos dispositivos e trabalha de maneira estável em redes com alta taxa de perdas. Por fim, AMQP não está preparado para aplicações com componentes restritos pela grande memória utilizada pelo dispositivo [Chaudhary et al. 2017].

A necessidade de recursos de processamento no concentrador foi estudada em [Torres et al. 2016]. Os autores mostram que como o concentrador deve gerenciar filas de pacotes, informações sobre a sessão com o dispositivo, entre outras, há uma exigência de recurso dedicado para o mesmo.

Diferentemente dos trabalhos acima, este trabalho utiliza um dispositivo com processamento realmente restrito, com poucos *KBytes* de memória e baixo poder de processamento, para análise de desempenho, replicando um cenário mais real de IoT. Além disso, para caracterizar um cenário mais realístico, considerou-se a influência do tráfego de fundo no desempenho dos protocolos MQTT e CoAP.

4. Avaliação qualitativa dos protocolos de aplicação

A partir dos trabalhos relacionados discutidos anteriormente, esta seção apresenta uma análise qualitativa dos requisitos para as aplicações de Internet das Coisas. Inicialmente, as aplicações IoT foram classificadas em três domínios: residencial, comercial e industrial, que são descritos a seguir.

Residencial As aplicações residenciais se subdividem em aplicações implementadas por entusiastas e as desenvolvidas por empresas [Brush et al. 2011]. As aplicações implementadas por entusiastas utilizam dispositivos de prototipagem e sensores disponíveis no mercado, tais como ESP8266, *Arduínos*, *Raspberries*, etc. Por outro lado, empresas disponibilizam projetos proprietários e, muitas vezes integram suas soluções com sistemas de grandes empresas. Essas soluções geralmente visam sistemas de automatização de ambientes inteligentes, tais como aquecimento ou resfriamento de ambientes, monitoramento de segurança, iluminação inteligente, e automatização de tarefas em horários específicos do dia.

Comercial Neste domínio estão os frameworks comerciais de desenvolvimento de aplicações IoT. Estes frameworks dão suporte aos domínios residenciais e industriais e normalmente, oferecem serviços como armazenamento e processamento de dados [Derhamy et al. 2015]. Desta forma, há uma camada de abstração onde as empresas, residências e indústrias se conectam e utilizam serviços das principais companhias tecnológicas, como *Microsoft*, *Google*, *IBM* e *Intel*, além das dezenas de outras plataformas que disputam o mercado [Mineraud et al. 2016]. De fato, produtos como *Azure IoT* da Microsoft, *Amazon AWS* da Amazon, *Google Cloud IoT* da Google trabalham com mais de um protocolo para gerenciar os dispositivos na borda da rede.

Industrial O domínio industrial trata de sistemas de tempo real e alta confiabilidade. No geral, estes sistemas possuem uma grande quantidade de sensores que precisam comunicar entre si e com um concentrador. Portanto, a utilização da rede de uso comum pode ser facilmente prejudicada devido à grande quantidade de informação que é trocada por eles [Da Xu et al. 2014]. Exemplos de aplicações no domínio industrial são os sistemas de monitoramento de pacientes (*eHealth*), sistemas anti-incêndio, controle e rastreamento de cargas em tempo real, etc.

Para analisar quais protocolos de aplicação IoT atendem cada um dos domínios acima, mapeamos como cada protocolo atende a um conjunto de requisitos, descritos a seguir. As análises se cada protocolo atende ou não cada requisito são baseadas nos experimentos realizados em [Bandyopadhyay and Bhattacharyya 2013], [Chaudhary et al. 2017], [De Caro et al. 2013], [Mun et al. 2016] e [Naik 2017], além das especificações dos protocolos. A Tabela 1 sumariza os resultados obtidos.

- **Restrição severa de processamento**: Este requisito refere-se a dispositivos com restrição de memória e processamento. Consideramos como restrição severa, dispositivos com menos de 80*KBytes* de memória e 80Mhz de processamento. Os protocolos MQTT, CoAP e MQTT-SN, possuem cabeçalhos pequenos o que ajuda a diminuir a necessidade de *buffers* e de dados processados. O AMQP possui ferramentas adicionais de transmissão e gerenciamento de mensagens o que necessita maior processamento.
- **Restrição energia**: Sensores em locais de difícil acesso precisam ter energia por longos períodos de tempo sem que seja necessária a troca de fontes energéticas.
- **Qualidade de sinal da rede**: A qualidade da rede utilizada para comunicação entre dispositivos em que ele utiliza para comunicação. A escala de respostas pode ser de *Ruim* (com mais de 20% de perda), *Média* (entre 0 e 20% de perda) e *Boa* onde não há perda.

- **Possibilidade de Concentrador**: Essa característica define se a estrutura de rede do local onde os dispositivos estão instalados permite ou necessita de um concentrador de dados. As repostas possíveis são *Sim* e *Não* e estão ligadas diretamente a estrutura de comunicação apresentada pelo protocolo. Aqueles que utilizam o modelo *Publisher/Subscriber* naturalmente necessitam de um concentrador.
- **Recursos do Concentrador**: Essa característica define se o concentrador requer recursos de processamento e memória para gerenciar os dados recebidos. A escala de respostas possíveis trata como *Baixa* quando há a possibilidade de utilizar outro dispositivo com baixo poder de processamento, *Média* quando esse concentrador por ser um *Raspiberry* ou equipamentos similares e *Alta* quando há a necessidade de mais processamento.
- **Capacidade de reconhecimento de outros dispositivos**: Essa característica define se, em uma comunicação M2M, o protocolo possui a capacidade de descobrir outros dispositivos, inclusive dispositivos concentradores [Villaverde et al. 2014]. As respostas representam se os protocolos possuem a característica de descoberta de outros dispositivos ativa.
- **Envio de dados em tempo real**: Define se um protocolo é recomendado para uso em situações em que o tempo real é importante. As respostas possíveis são *Sim* e *Não*.
- **Tamanho da carga de dados**: Essa característica mostra se o protocolo está preparado para o envio de mensagens com grandes quantidades de informação (alto *payload*). Esta característica pode ocorrer tanto no nível do protocolo de aplicação ou pode ser implementada a nível da camada de transporte.
- **Confiabilidade na entrega dos dados**: Essa característica define se o protocolo consegue uma entrega confiável de dados para uma aplicação que exija tal requisito. Os protocolos CoAP, MQTT-SN e AMQP possuem seus próprios sistemas de confiabilidade sobre as camadas de transporte TCP e UDP. O MQTT possui nos níveis de QoS 1 e 2, porém no nível mais comum de utilização, o QoS 0, o protocolo deixa esses controles para a camada de transporte.
- **Interoperabilidade**: Essa característica define a possibilidade de interação com plataformas já disponíveis por grandes empresas. O AMQP possui essa característica como fundamento.
- **Envio constante de informações**: Essa característica define a capacidade do protocolo de enviar mensagens de modo constante.

Considerando os requisitos apresentados acima, o MQTT apresenta maior estabilidade devido a utilização do TCP. Aplicações que necessitem troca de mensagens com grande quantidade de dados em redes locais também podem utilizar o MQTT. O MQTT também possui implementações disponíveis para diversos dispositivos, incluindo versões para dispositivos com baixo poder de processamento o que aumenta ainda mais as possibilidades de utilização. Além disto, as nuvens de grandes empresas, como *Amazon* e *Microsoft* fornecem suporte ao MQTT.

Por outro lado, o protocolo de aplicação CoAP apresenta vantagens em todos os domínios. A arquitetura cliente/servidor permite que os dispositivos economizem energia. Além disso, o uso do UDP torna o fluxo de dados menor. Devido a semelhança com o padrão HTTP, o protocolo CoAP tem grande adoção entre desenvolvedores Web.

Tabela 1. Comparação Entre os Protocolos Estudados

Características	MQTT	MQTT-SN	CoAP	AMQP
Restrição severa de processamento	Sim	Sim	Sim	Não
Restrição severa de energia	Não	Sim	Sim	Não
Qualidade do sinal da rede	Boa	Ruim	Ruim	Boa
Possibilidade de concentrador	Sim	Sim	Não	Sim
Capacidade de recursos do concentrador	Média	Média/Baixa	–	Alta/Média
Capacidade de reconhecimento de outros dispositivos	Não	Sim	Sim	Não
Envio em tempo real de dados	Não	Não	Sim	Não
Comunicação com dados muito grandes	Sim	Não	Não	Sim
Confiabilidade na entrega dos dados	Sim	Sim	Não	Sim
Interoperabilidade entre fabricantes	Não	Não	Não	Sim
Envio constante de informações	Sim	Sim	Sim	Sim

5. Análise quantitativa

O MQTT e o CoAP são protocolos com arquiteturas distintas e com ampla adoção da indústria. Por este motivo, este trabalho analisa o desempenho de ambos quando utilizado com dispositivos com recursos restritos de processamento e memória. Para cada protocolo, foram realizados dois experimentos, sendo que cada experimento foi repetido com e sem tráfego de fundo na rede.

Monitoramento em tempo (quase) real: O intervalo mínimo entre mensagens no ESP8266 é de 100ms. Por isto, avaliou-se o monitoramento em tempo quase real com envio pacotes de 30 *bytes* com intervalo entre 100 e 900 milissegundos. Os testes duraram 10 minutos. O conteúdo das mensagens era composto por um identificador do dispositivo, um identificador da mensagem e por um *timestamp*.

Variação no tamanho da mensagem: foram enviados pacotes de tamanho 10, 100, 500 e 1000 *bytes* em intervalos de 1, 5, 10, 30 e 60 segundos durante 1 hora.

A topologia do *testbed* pode ser vista na Figura 1. O ESP8266, representando dispositivos IoT, possui um processador 32-bit RISC com 80 MHz de *clock* e 80*KBytes* de memória RAM, sendo que aplicações podem utilizar apenas cerca de 50 *Kbytes*. Foram utilizadas as bibliotecas [Kovalenko 2018] para o MQTT e [Poornima Nagesh 2018] para o CoAP[1]. O servidor/concentrador captura as informações da rede com o *Wireshark*.

[1] As implementações estão disponíveis em https://gitlab.com/vmota/mqtt-coap-esp

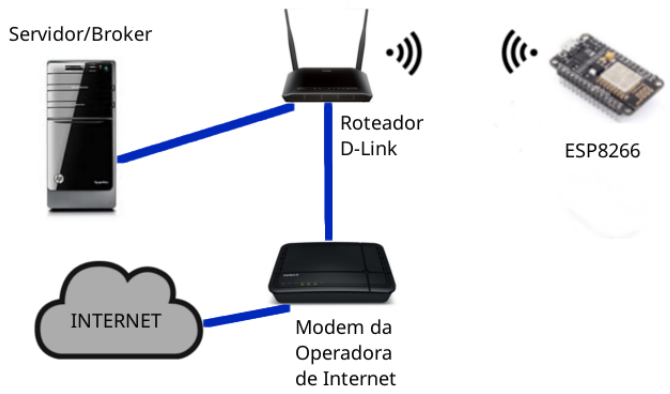


Figura 1. Topologia Utilizada nos Testes

No servidor, o concentrador MQTT *Mosquito Broker* e um servidor CoAP estavam instalados. Para os testes sem tráfego de fundo, isolava-se a rede e apenas o ESP8266 e o servidor estavam conectados a rede sem fio. Já no teste com tráfego de fundo, permitia-se a utilização do roteador sem fio por outros usuários. Foram mensurados quatro métricas:

Taxa de entrega: Quantidade de mensagens recebidas pelo servidor por segundo.

Tempo médio entre recebimento de pacotes: Mostra a média do tempo entre captura dos pacotes no *Broker* e foi utilizada para o MQTT. A média é calculada pela equação:

$$T_{dm} = \frac{\sum_{i=0}^{N-1} t_{i+1} + t_i}{N_{PUB}}$$

sendo, t_i o tempo de recebimento da mensagem *i*, t_{i+1} o tempo da próxima mensagem e N_{PUB} o número de mensagens publicadas.

Tempo médio de transmissão: Trata do tempo médio entre o envio e o recebimento da mensagem no protocolo CoAP, ou seja, o *Round Time Trip* (RTT). O tempo médio é dado pela equação:

$$T_m = \frac{\sum_{i=0}^{N-1} \frac{t_{ACK} - t_{GET}}{2}}{N_{ACK}}$$

sendo, t_{ACK} o tempo em que a mensagem de confirmação de recebimento foi capturada, t_{GET} o tempo em que a mensagem de requisição foi enviada e N_{ACK} o número de mensagens de confirmação.

Número de retransmissão dos pacotes: número de pacotes que foram retransmitidos até que a mensagem fosse recebida pelo computador.

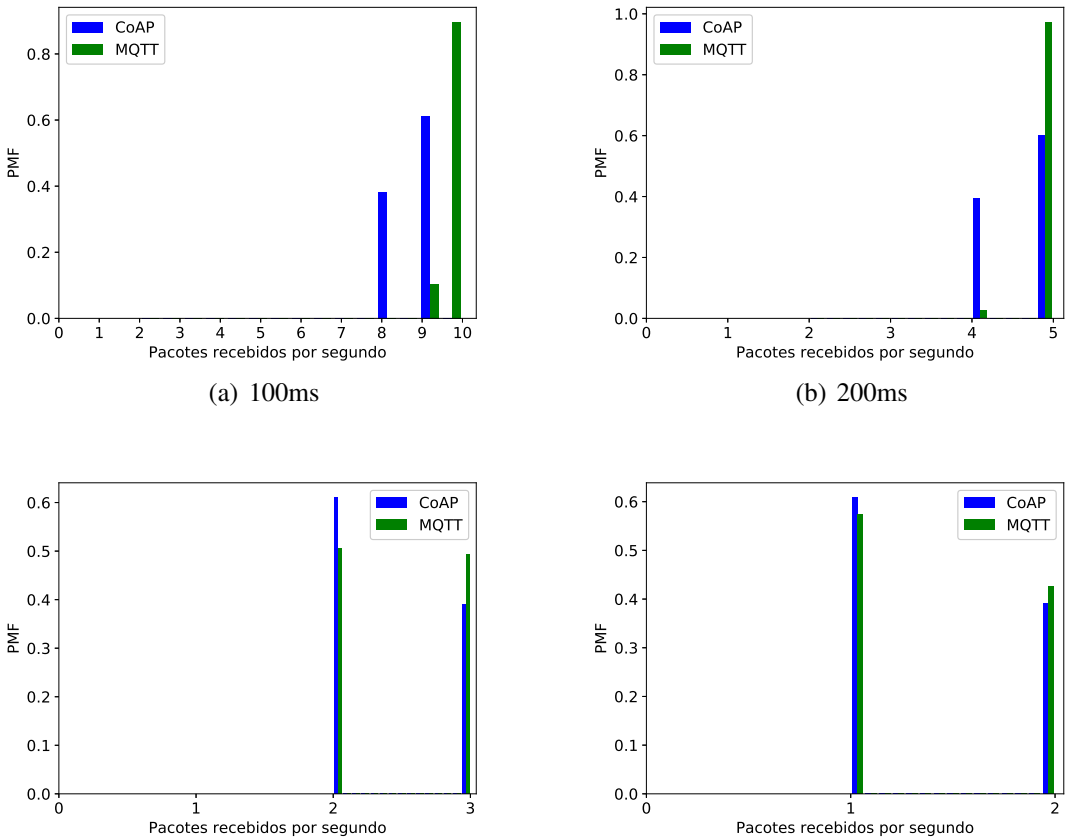
6. Resultados e Análises

6.1. Monitoramento em tempo (quase) real

Consideramos como monitoramento em tempo quase real aplicações que enviem mensagens com uma taxa de envio menor que um segundo. A primeira métrica que analisamos foi a taxa de entrega.

Para calcular a taxa de entrega, o dispositivo enviava mensagens entre intervalos de 100, 400 e 700 milissegundos. Cada mensagem recebe um identificador inteiro sequencial, o servidor verifica se está recebendo as mensagens na sequência. Em um cenário ideal, o servidor irá receber 10 pacotes por segundo quando o envio tem intervalos for de 100 ms, 2 ou 3 pacotes para intervalos de 400ms e 1 ou 2 pacotes para intervalos de 700ms. Ressalta-se que devido as limitações de hardware do ESP8266, não foi possível enviar mensagens em intervalos inferiores a 100ms.

A Figura 2 apresenta a função massa de probabilidade (*probability mass function* - PMF) dos pacotes recebidos pelo servidor/concentrador a cada segundo em uma rede sem tráfego de fundo.



(a) 100ms (b) 200ms

(c) 400ms (d) 700ms

Figura 2. Função massa de probabilidade (PMF) de pacotes recebidos por segundo nos experimentos sem tráfego de fundo dos protocolos CoAP e do MQTT, com envio de mensagens entre 100 e 700 milissegundos.

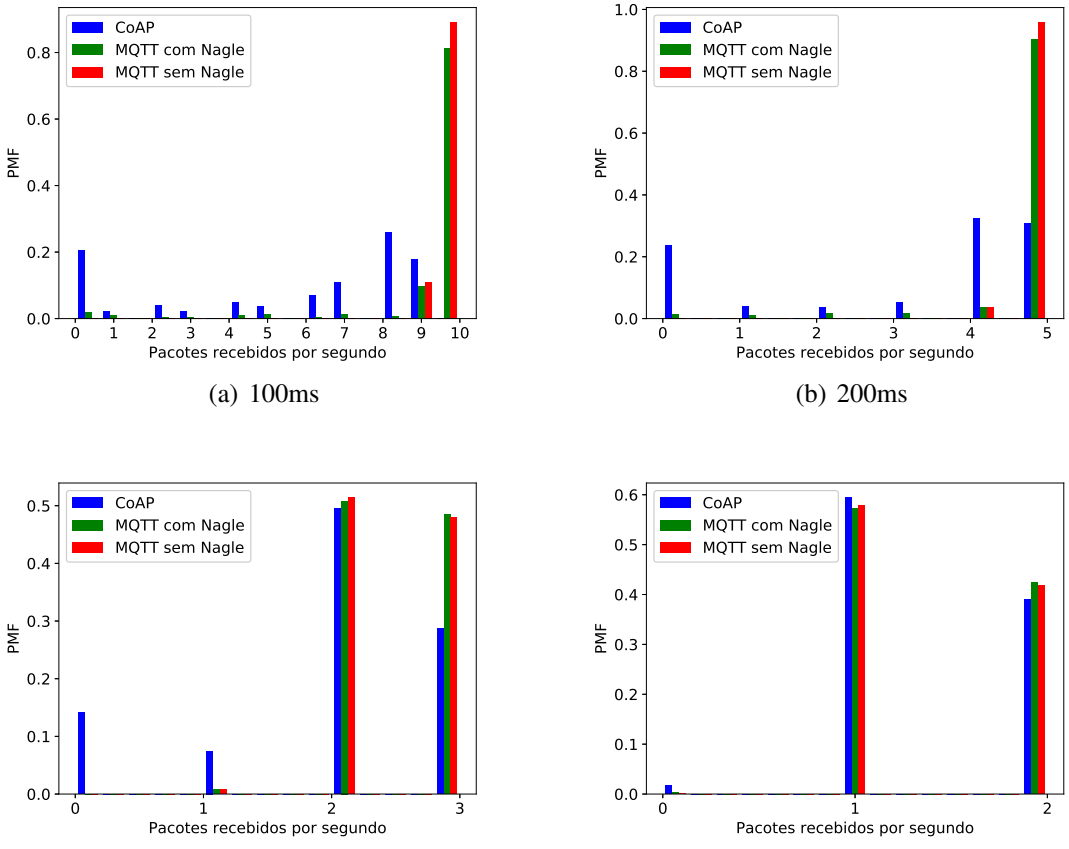
Observa-se que no cenário de intervalo entre mensagens de 100ms, Figura 2(a), o CoAP não conseguiu entregar 10 mensagens por segundo ao servidor. Como o CoAP utiliza o UDP, não exige confiabilidade na entrega, os pacotes esperam no *buffer* e/ou podem ser descartados.

Para o protocolo MQTT, quanto menor o intervalo entre envios de mensagens, maior a chance de o pacote enviado pelo dispositivo não chegar ao *Broker*. Mesmo utili-

zando o TCP na camada de transporte, observou-se que não há recebimento de mensagens em alguns instantes durante a captura de pacotes.

A explicação para o não recebimento da totalidade de mensagens pelo servidor refere-se ao algoritmo de Nagle. implementado por padrão no *lwIP*. O algoritmo de Nagle pode interromper, por um período de tempo, o envio de mensagens quando uma destas não recebe um ACK. Assim, todas as outras mensagens que poderiam ser entregues permanecem no *buffer* até que ele receba um ACK ou até que o tamanho combinado no *handshake* do TCP seja atingido.

Nos experimentos, o efeito do algoritmo pode ser percebido no *Wireshark* com o recebimento de uma mensagem de erro *"TCP Spurious Retransmission"*, seguida de um pacote encapsulando mais de uma mensagem do MQTT.



(a) 100ms (b) 200ms

(c) 400ms (d) 700ms

Figura 3. Função massa de probabilidade (PMF) de pacotes recebidos por segundo nos experimentos com tráfego de fundo dos protocolos CoAP, MQTT e MQTT sem Nagle, com envio de mensagens entre 100 e 700 milissegundos.

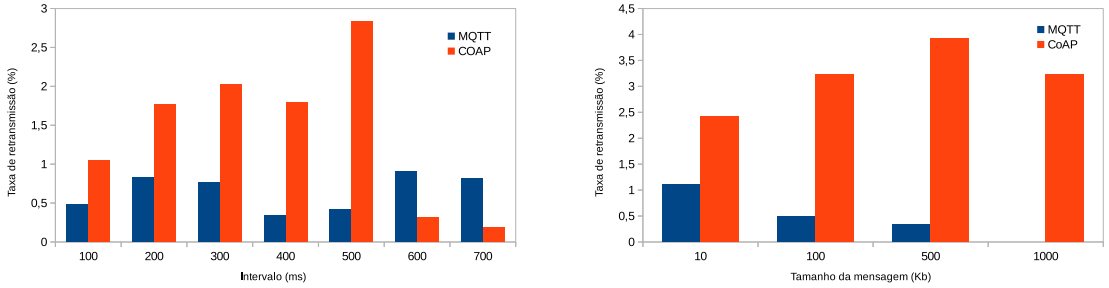
A Figura 3 apresenta a PMF do recebimento de pacotes quando existe um tráfego de fundo. O tráfego de fundo foi gerado por outros dispositivos conectados ao mesmo ponto de acesso para navegação web, vídeos, músicas, etc. Além disso, neste caso também analisamos os efeitos de habilitar e desabilitar o algoritmo Nagle. O desempenho do MQTT com o algoritmo Nagle desabilitado se assemelha ao desempenho do

MQTT sem tráfego de fundo. Isso acontece por que não há a retenção das mensagens e o fluxo de dados consegue seguir mesmo com perda de mensagens.

Comparando os dois protocolos, observa-se que o MQTT consegue superar o CoAP para intervalos menores que 500 milissegundos. Depois desse valor, até o limite de novecentos milissegundos, a taxa de entrega fica muito próxima.

Em relação ao *número de mensagens retransmitidas*, no MQTT calcula-se o número de mensagens *"TCP Spurious Retransmission"*, causadas pelo controle do algoritmo de Nagle. As retransmissões no CoAP acontecem quando o cliente não recebe resposta e faz uma nova requisição.

A Figura 4(a) apresenta a taxa de retransmissão (mensagens retransmitidas por total de mensagens) do CoAP e do MQTT. Observa-se que o protocolo MQTT possui um número menor de retransmissão para intervalos de envio menores que 500ms, que era de se esperar pela confiabilidade do TCP em relação ao UDP. Ao aumentar o intervalo entre envios o CoAP diminui o número de erros, porém o MQTT passa a ser influenciado pelo algoritmo de Nagle, implementado internamente no TCP.



(a) Intervalo entre envio menor que 1s.

(b) Tamanho da mensagem

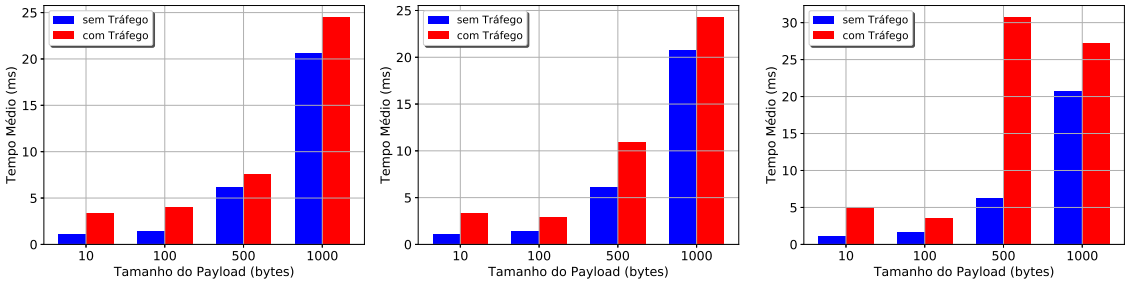
Figura 4. Retransmissões nos protocolos CoAp e MQTT

6.2. Tamanho da mensagem - Payload

O tamanho da carga de dados em cada protocolo foi variado entre {10, 100, 500, 1000} *bytes*. A biblioteca utilizada para o MQTT conseguiu ser mais estável e enviar mensagens de até 1500 *bytes* sem que utilizasse toda a memória RAM do ESP. Já para o CoAP, após a alteração da biblioteca, foi possível enviar até 1024 *bytes* antes que o mesmo efeito acontecesse. De fato, como mostra a Figura 4(b), o número de retransmissões no MQTT diminuiu com o crescimento do tamanho do *payload*. Justificado pelo gerenciamento de *buffer* que ocorre no MQTT.

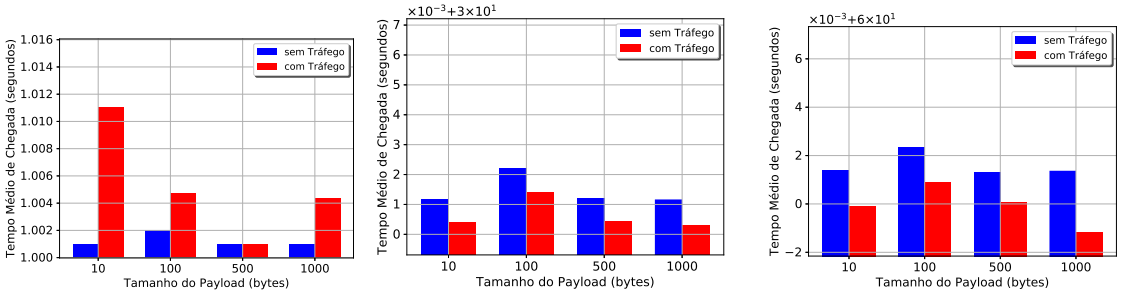
Tempo médio de transmissão em relação ao tamanho da mensagem para o protocolo CoAP é mostrado na Figura 5. CoAP se mostrou vulnerável ao tráfego de fundo sendo que situações de retransmissão de pacotes com mais de três tentativas aconteceram mais de uma vez o que aumentou consideravelmente o tempo da média.

Para o MQTT, foi calculado o tempo médio de recepção entre mensagens, visto que não é possível inferir o tempo de envio da mensagem. De modo geral, o tamanho do *payload* não interfere na geração e envio de mensagens, como mostra a Figura 6.



(a) 1s (b) 30s (c) 60s

Figura 5. Tempo médio (RTT) requisições-resposta do CoAP com e sem tráfego de fundo, com envio de mensagens entre 1 e 60 segundos



(a) 1s (b) 30s (c) 60s

Figura 6. Tempo entre recebimento de pacotes do MQTT com e sem Tráfego de fundo e envio de mensagens entre 1 e 60 segundos.

Entretanto, é possível notar que a diferença média de tempos sem tráfego de rede é maior que a média quando há tráfego de rede, excluindo 6a.

A Figura 6(a) mostra que no momento da captura aconteceram algumas perdas de pacotes na camada TCP e foram feitas algumas retransmissões. Ocorreu também o efeito do algoritmo de Nagle em alguns momentos e o ESP8266 parou de enviar pacotes por uns três a oito segundos influenciando na média dos resultados.

Embora uma comparação direta entre os protocolos não pode ser devida as métricas serem calculadas de forma distinta, nota-se que os dois tiveram problemas ao utilizar uma rede com tráfego de fundo. Observa-se que o número de mensagens retransmitidas pelo CoAP foi maior para todos os tamanhos de carga de dados na mensagem, Figura 5. O CoAP teve um aparente aumento do tempo de resposta com o aumento do *payload*. O MQTT teve um desempenho mais uniforme comparado com os outros tamanhos. Vale ressaltar que o TCP estava fragmentando os pacotes com tamanho máximo de 536 *bytes* que é implementado como padrão no *lwIP* disponível no ESP8266. Já o CoAP, estava mandando o pacote inteiro como característica do UDP, tendo como máximo valor 1024 *bytes* devido à limitação do dispositivo.

7. Conclusão

Este trabalho analisou qualitativamente e quantitativamente os principais protocolos da camada de aplicação para Internet das Coisas. Qualitativamente, observou-se que o

MQTT, por sua arquitetura *publish/subscribe* foi adotado por diversas plataformas comerciais de armazenamento de dados IoT. A versão MQTT-SN provê maior versatilidade devido a utilização de UDP no lugar do TCP como camada de transporte. O CoAP é um dos mais versáteis, tendo vantagens em todos os cenários pelo seu modelo de comunicação e compatibilidade com HTTP. Já o AMQP apresenta melhores mecanismos de controle de fluxo e erro, acrescentando confiabilidade e melhorando a interoperabilidade.

Para realizar os testes quantitativos, foi utilizado o SoC ESP8266 e as bibliotecas públicas do CoAP e MQTT. Diferentemente de outras comparações realizadas na literatura, estes dispositivos não suportam o envio de mensagens com payload maiores que 1024 *bytes* e nem o envios de mensagens com intervalos menores do que 100ms. Além disso, a implementação padrão do TCP sobre o lwIP, utilizado nestes dispositivos, habilita o algoritmo Nagle por padrão, para reduzir o número de segmentos muito pequenos enviados, e assim, economizar energia. Isso afeta o desempenho do MQTT para funcionar como um protocolo de tempo-real. Considerando que as bibliotecas públicas continham bugs e/ou incompatibilidades, este trabalho tornou público para desenvolvedores as implementações dos protocolos de aplicação para IoT, CoAP e MQTT.

Quantitativamente, observou-se que o protocolo CoAP foi mais sensível ao aumento do *payload* do que o MQTT nos testes com tráfego de fundo na rede, principalmente com o algoritmo de Nagle desabilitado. Portanto, o MQTT mostrou-se mais estável que o CoAP ao utilizar um dispositivo restrito. Além disso, o tráfego de fundo causou retransmissão de mensagens em ambos os protocolos. Como o CoAP utiliza o UDP na camada de transporte, o CoAP implementa seu próprio mecanismo de controle de erros. Contudo, esse mecanismo pode gerar problemas no envio constante de mensagens, pois o modelo de retransmissão do CoAP pode fazer com que o mesmo aguarde o recebimento de confirmação antes de enviar novas mensagens.

Por fim, espera-se que este trabalho possa auxiliar desenvolvedores a escolherem qual o melhor protocolo da camada de aplicação baseado nos requisitos apresentados. Como trabalhos futuros, pretende-se implementar as especificações dos protocolos MQTT-SN e AMQP para dispositivos de processamento e memória restritos.

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.

Appel, S., Sachs, K., and Buchmann, A. (2010). Towards benchmarking of AMQP. *Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems, DEBS 2010*, (October 2016):99–100.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

Bandyopadhyay, S. and Bhattacharyya, A. (2013). Lightweight internet protocols for web enablement of sensors using constrained gateway devices. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 334–340. IEEE.

Brush, A., Lee, B., Mahajan, R., Agarwal, S., Saroiu, S., and Dixon, C. (2011). Home automation in the wild: challenges and opportunities. In *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2115–2124. ACM.

Chaudhary, A., Peddoju, S. K., and Kadarla, K. (2017). Study of internet-of-things messaging protocols used for exchanging data with external sources. pages 666–671.

Da Xu, L., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243.

De Caro, N., Colitti, W., Steenhaut, K., Mangino, G., and Reali, G. (2013). Comparison of two lightweight protocols for smartphone-based sensing. In *Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on*, pages 1–6. IEEE.

Derhamy, H., Eliasson, J., Delsing, J., and Priller, P. (2015). A survey of commercial frameworks for the Internet of Things. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2015-Octob.

Kovalenko, O. (2018). Arduinomqtt: Mqtt client library for arduino based on the eclipse paho project.

Mazzer, D., Frigieri, E., and Parreira, L. (2015). Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparação de Abordagens. *Inatel.Br*.

Mineraud, J., Mazhelis, O., Su, X., and Tarkoma, S. (2016). A gap analysis of internet-of-things platforms. *Computer Communications*, 89:5–16.

Mun, D.-H., Le Dinh, M., and Kwon, Y.-W. (2016). An assessment of internet of things protocols for resource-constrained applications. 1:555–560.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*.

Poornima Nagesh, L. P. (2018). Esp-coap: This is a arduino library for the esp8266 12e.

Shelby, Z., Hartke, K., and Bormann, C. (2014). The constrained application protocol (coap). RFC7252.

Torres, A. B., Rocha, A. R., and de Souza, J. N. (2016). Análise de desempenho de brokers mqtt em sistema de baixo custo. In *Anais do XXXVI congresso da sociedade brasileira de computaç ao. Sociedade Brasileira de Computaçao*.

Truong, A. S.-C. and Linh, H. (2013). MQTT For Sensor Networks (MQTT-SN) Protocol Specification. *International Business Machines Corporation (IBM*, 1:28.

Villaverde, B. C., de Paz Alberola, R., Jara, A. J., Fedor, S., Das, S. K., and Pesch, D. (2014). Service discovery protocols for constrained machine-to-machine communications. *IEEE Communications Surveys and Tutorials*, 16(1):41–60.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32.