Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparação de Abordagens

Daniel Mazzer and Edielson Prevato Frigieri
Instituto Nacional de Telecomunicações - Inatel
Santa Rita do Sapucaí - MG - Brasil
{daniel.mazzer, edielson}@inatel.br

Luís Felipe Costa Gambogi Parreira
Instituto Nacional de Telecomunicações - Inatel
Santa Rita do Sapucaí - MG - Brasil
lfgambogi@gmail.com

***Resumo*— O movimento da Internet das Coisas abre novas possibilidades para serviços e negócios com novos desafios tecnológicos como eficiência energética, operação em ambientes limitados, segurança e privacidade. Com a expectativa do grande número de dispositivos conectados à Internet do Futuro, assume-se que a escalabilidade também será um desafio. Para endereçar essas limitações, alguns protocolos estão sendo propostos. Nesse artigo, alguns deles serão apresentados e comparados qualitativamente, resumindo suas principais características e limitações e destacando os melhores cenários onde cada abordagem tem melhor desempenho.**

Palavras-chave*— *Ambientes limitados, Internet das Coisas, Máquina a Máquina, Segurança, Escalabilidade.

I. INTRODUÇÃO

A possibilidade de conectar pessoas e objetos inteligentes através de uma infraestrutura comum — a Internet —, tem sido o foco de muitas pesquisas recentes em Tecnologias da Informação e Comunicação (TIC). Essa nova abordagem recebe o nome de Internet das Coisas (*Internet of Things* — IoT), termo que foi usado pela primeira vez por Ashton em 1999 [1]. A ideia principal se refere a uma rede unificada para interconectar pessoas a qualquer tipo de *coisa*, que pode ser um objeto real ou virtual, *e.g.*, um dispositivo de hardware ou um *web service* [2].

Esse cenário usa a Internet para promover a troca de informações e comunicação para realizar diferentes tipos de serviços como monitoramento, rastreamento, localização e reconhecimento inteligente [3]. Assim, novos modelos de cenários de negócios podem ser criados, por exemplo, provedores de serviços de informação, Produtos como Serviço (*Products-as-a-Service* — PaaS), envolvimento de usuário final, análise de hora certa para negócios, tomada de decisão e outros [4].

Junto das novas possibilidades, aparecem novos desafios como eficiência energética e ambientes limitados (dispositivos, larguras de banda, redes, etc.) [5]–[7] e preocupações quanto a segurança e privacidade [8]. Se essa nova tecnologia não garante a segurança da informação privada, os usuários serão resistentes à sua adoção em seu ambiente e vida [9].

Uma vez que se espera que essa mudança de tecnologia seja maior que a causada pelo advento dos telefones celulares, problemas sérios de escalabilidade podem ser destacados no contexto de protocolos máquina a máquina (*machine-to-machine* — M2M) padronizados enquanto a interação homem-máquina é facilitada [10]. A maior preocupação é que muitos protocolos foram desenvolvidos para ambientes ilimitados, onde o número de dispositivos é limitado às centenas ou milhares. Entretanto, muitos cenários IoT no contexto da Internet do Futuro apresentam ambientes muito limitados em recursos com milhões ou até bilhões de dispositivos. Essas preocupações estão motivando o projeto de novos protocolos focados em ambientes limitados em termos de rede e recursos computacionais. Alguns deles são mais adequados para sistemas que demandem apenas algumas transferências de bytes por dia e possam ser executados em dispositivos alimentados por baterias ao longo de anos. Outros são capazes de transportar pacotes de Linguagem de Marcação Extensível (*Extensible Markup Language* — XML) e garantir uma melhor e mais complexa entrega de mensagens. Para escolher o melhor protocolo, uma análise cuidadosa da aplicação-alvo deve ser feita, assim como das características dos protocolos e dos requisitos.

Seguindo essas tendências, o objetivo desse trabalho é apresentar, analisar qualitativamente e discutir alguns dos protocolos M2M propostos, identificando suas principais características e limitações, além de destacar os melhores cenários onde cada um pode ser aplicado.

Para isso, o restante do artigo é organizado da seguinte forma: a Seção II apresenta alguns protocolos M2M para cenários IoT; a Seção III os discute, resumindo suas principais características e limitações, e finalmente, a seção IV traz as conclusões do artigo.

II. PROTOCOLOS M2M PARA A INTERNET DAS COISAS

Há vários protocolos propostos para comunicação M2M com foco em ambientes limitados. A maioria é baseada no protocolo IP com foco em conexões UDP devido às limitações das redes às quais são aplicados. Dentre eles, o MQTT (*Message Queue Telemetry Transport*) [11], o CoAP (*Constrained Application Protocol*) [12] e o AMQP (*Advanced Message Queuing Protocol*) [13] são frequentemente empregados. Eles são populares nos produtos comerciais com grande disponibilidade de *web services*, além de terem despertado o interesse da comunidade de código aberto [14], [15]. A vantagem de se usar um protocolo aberto é

a possibilidade do desenvolvedor focar o negócio da aplicação, deixando que a entrega de mensagens fique a cargo do protocolo. Além disso, esses protocolos são adequados para uso em hardware com baixa capacidade de memória e microcontroladores com baixa capacidade de processamento [16], [17].

A. *MQTT*

Criado pela IBM e pela Eurotech, o MQTT é um protocolo aberto projetado para ser simples, leve e de fácil implementação: características adequadas para dispositivos embarcados com recursos de memória e/ou processamento limitados. O pequeno *overhead* de transporte (cabeçalho de tamanho fixo de 2 bytes) faz do MQTT uma solução interessante para redes não-confiáveis com recursos limitados, como largura de banda e alta latência [11]. Esse protocolo é baseado em um *broker* usando o padrão de mensagens publica/assina, enquanto o *broker* do servidor atua como um intermediário para mensagens enviadas de um dispositivo que publica para clientes assinantes, provendo uma distribuição de mensagens um-para-muitos desacoplada do caso de uso da aplicação. Todas as mensagens endereçadas para um tópico específico[1], enviadas por uma entidade que publica, será entregue, pelo *broker* do servidor, para os assinantes daquele tópico, como representado na Fig. 1.

O MQTT V3.1 suporta 3 níveis de Qualidade de Serviço (QoS) que representam a confiabilidade de entrega das mensagens [11]. A **Erro! Fonte de referência não encontrada.** mostra a troca de pacotes de acordo com esses 3 níveis de QoS diferentes. Na **Erro! Fonte de referência não encontrada.** (a), o nível 0 é usado e quem publica envia a mensagem no máximo uma vez e não verifica se a mensagem chegou ao destino. Esse menor nível é chamado de "fire-and-forget" (disparar e esquecer) e a mensagem pode ser perdida dependendo das condições da rede.

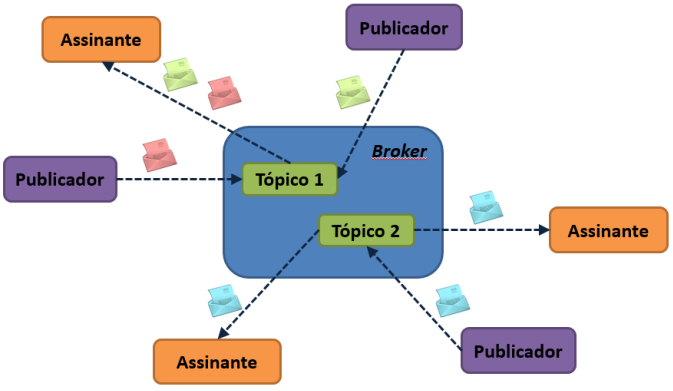

Fig. 1. Modelo de mensagens publica/assina.

O nível 1 de QoS é ilustrado na **Erro! Fonte de referência não encontrada.** (b). Também é chamado de entrega reconhecida. Quem publica envia uma mensagem pelo menos uma vez e verifica o estado da entrega usando a mensagem de verificação de estado PUBACK. Entretanto, se o PUBACK se perde, o *broker* do servidor pode possivelmente enviar a mesma mensagem duas vezes, uma vez que não houve confirmação de a mensagem ter sido entregue. No nível de QoS 2 mostrado na **Erro! Fonte de referência não encontrada.** (c), as mensagens são entregues exatamente uma vez usando um *handshake* em 4 vias. Esse modelo também é chamado de entrega garantida. Devido ao seu processo complicado, é possível haver atrasos fim-a-fim maiores, mas não há mensagens perdidas nesse nível. Quanto maior o nível de QoS, maior é a troca de pacotes. Se a perda de mensagens não é um problema, um menor nível de QoS pode ser usado, resultando em menor consumo da banda disponível e menor atraso fim-a-fim [18], o que representa redes limitadas cabeadas ou sem fio. Para reduzir ainda mais o uso da banda, o UDP pode ser usado em vez do TCP, porém com redução da garantia de entrega das mensagens.

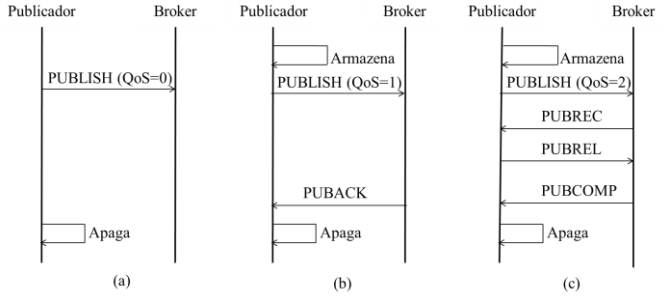

Fig. 2. Diferentes níveis de QoS para o modelo de mensagens Publish/subscribe

O protocolo MQTT não foi projetado considerando segurança e, como muitos outros protocolos baseados no TCP, usa o *Secure Sockets Layer* (SSL) ou *Transport Layer Security* (TLS) para segurança. Quando uma mensagem CONNECT é enviada para o *broker*, um nome de usuário e senha podem ser usados para autenticação. Porém, como destacado por Collina *et al* (2012), as credenciais de nome de usuário e senha são transmitidas sem criptografia, causando um dos problemas de segurança do protocolo. Entretanto, um ponto importante é que o MQTT não tem conhecimento dos dados que transporta, e assim os dados úteis podem ser criptografados em algum nível para aumentar a segurança da comunicação.

Apesar dos problemas de segurança, algumas características importantes podem ser destacadas:

- Mensagem de *keep-alive* (PINGREQ, PINGRESP), com a qual o *broker* pode detectar a desconexão de um cliente mesmo que ele não envie explicitamente uma mensagem DISCONNECT.
- Retenção de mensagem, em que uma mensagem de PUBLISH em um tópico específico pode ser retida no *broker*, permitindo que um assinante recém-conectado no mesmo tópico possa receber a mensagem.
- Mensagem de intenção (especificada na mensagem de CONNECT com tópico, QoS e retenção), permite que clientes assinantes sejam informados de uma

[1] Toda mensagem MQTT inclui um tópico que a classifica. Os *brokers* MQTT usam tópicos para determinar quais assinantes devem receber mensagens publicadas no *broker*.

desconexão de cliente inesperada.

- Assinatura durável que retém todas as assinaturas no *broker* quando um cliente é desconectado e permite que sejam recuperadas quando o cliente se reconectar.

Essas características do MQTT levaram à sua aplicação a soluções onde o baixo consumo de bateria é um pré-requisito e onde há pouca largura de banda disponível ou conexão intermitente [19]–[21], o que caracteriza o primeiro nível em uma aplicação de redes de sensores. Essas características e usos encorajaram a criação de uma nova versão do padrão do MQTT chamada MQTT-SN [22], com foco em redes de sensores. Essa versão foi desenvolvida para ser executada em redes diferentes das TCP/IP, como 6LoWPAN, protocolos serializados customizados ou protocolos de radiofrequência customizados baseados no padrão IEEE 802.15.4, por exemplo, o ZigBee®. Uma das principais diferenças entre os dois padrões, além da camada de rede em que focam, é a simplificação das mensagens trocadas entre *broker* e clientes, usando identificadores de tópicos predefinidos e nomes curtos de tópicos além de um formato de mensagens curto [7].

B. CoAP

O protocolo CoAP foi projetado pelo Grupo de Trabalho para Ambientes RESTful Limitados (CoRE) da Força-Tarefa de Engenharia da Internet (IETF). Adaptado do HTTP, foi otimizado para dispositivos com potência e capacidade de processamento limitados e é geralmente aplicado a objetos inteligentes em ambientes IoT [12]. Executado sobre o protocolo de transporte UDP, o CoAP especifica um conjunto mínimo de requisições REST incluindo POST, GET, PUT e DELETE, com suporte a armazenamento de recursos e descoberta de recursos embutida.

O CoAP adota o modelo requisição/resposta, onde cada dispositivo atua como um cliente ou um servidor e os recursos podem ser acessados por URIs. Diferente do HTTP, a conexão não é estabelecida antes da troca de mensagens. A comunicação acontece de forma assíncrona. Há quatro tipos de mensagens: CON (Confirmável), NON (Não-Confirmável), ACK (Reconhecimento) e RESET. As mensagens CON e NON atuam como níveis de QoS, como exemplificado na Fig. 2. Uma requisição enviada usando o tipo NON não tem mensagem de reconhecimento enviada de volta pelo receptor, o que caracteriza um baixo nível de QoS. Esse tipo de troca de mensagens é ilustrado na Fig. 3.

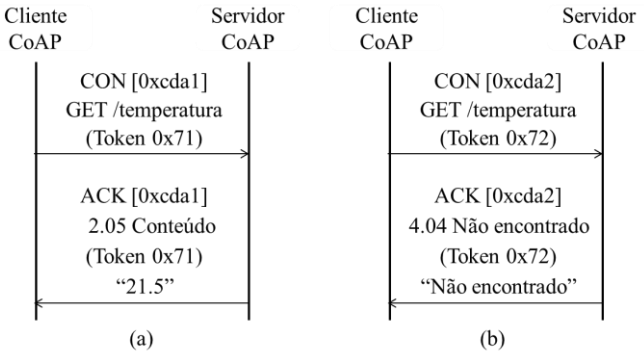

Fig. 2. Duas requisições GET com respostas interceptadas [12]: (a) Acesso bem-sucedido ao recurso e (b) Recurso não encontrado.

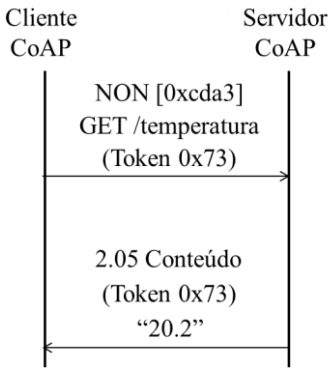

Fig. 3. Requisição e resposta de uma mensagem não-confirmável [12].

Para garantir a segurança durante a troca de mensagens entre cliente e servidor, o CoAP usa o protocol DTLS (*Datagram Transport Layer Security*), que é baseado no TLS sobre UDP em vez do TCP. O DTLS tem problemas de segurança, bem como o TLS. Um problema está relacionado a realizar a tradução DTLS quando o mapeamento CoAP é usado em um *proxy* para prover uma conexão fim-a-fim segura. Outro problema está relacionado com comunicações multiponto seguras que ainda não são suportadas [23].

Em comparação com o HTTP, o CoAP é mais eficiente em termos de custo porque realiza menor troca de dados entre o cliente e o servidor, resultando em menor consumo de potência ao usar equipamentos mais baratos em ambos os lados da conexão [24], [25]. Em resumo, as seguintes principais características do CoAP podem ser enumeradas:

- Um cabeçalho binário compacto combinado com o transporte baseado em UDP que reduz o *overhead* e, consequentemente, reduz atrasos e minimiza o uso da bateria durante transmissões;
- O suporte ao envio de informação assíncrona (opção observar) que permite que objetos inteligentes enviem informações sobre recursos apenas quando há mudanças. O dispositivo pode permanecer em modo *sleep* na maior parte do tempo, o que significa uma redução do consumo de potência. Um exemplo dessa operação pode ser visto na Fig. 4;
- O uso de um subconjunto mínimo das requisições REST permite o uso de hardware com menos recursos quando

comparado com o HTTP.

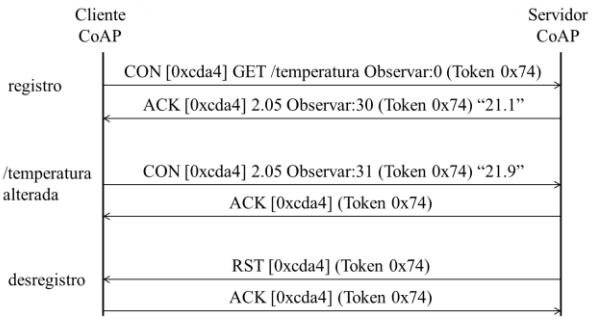

Fig. 4. Exemplo de troca de mensagens para a operação na opção observar.

Essas características tornaram o CoAP interessante para soluções que usem dispositivos embarcados com severas restrições de alimentação e memória, além de redes limitadas. A arquitetura REST e a fácil tradução para HTTP permitem a criação de cenários onde antigos clientes *web* podem acessar servidores CoAP de forma transparente usando *proxies* que disponibilizam os recursos CoAP como URIs comuns *http://* e *https://* [26]. Um cenário de exemplo é mostrado na Fig. 5.

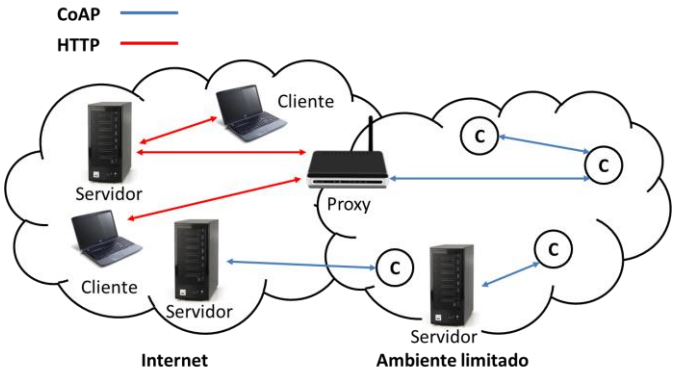

Fig. 5. Cenário *web* com HTTP e CoAP [26].

C. AMQP

O AMQP é uma especificação de padrão aberto de mensagens de *middleware*, baseado no paradigma de filas de mensagens orientadas a tópicos, em que produtos escritos para diferentes plataformas e em diferentes linguagens podem trocar mensagens. Esse protocolo é suportado por empresas-chave, incluindo Cisco Systems, Credit Suisse, Deutsche Borse Systems, Goldman Sachs, JPMorgan Chase Bank, Red Hat e 29West [27]. Apesar de ser um protocolo padronizado, nem todas as implementações são totalmente compatíveis com o padrão. Isso ocorre porque nem toda aplicação requer todas as funcionalidades providas pelo padrão, economizando tempo de desenvolvimento e reduzindo a demanda de recursos de hardware. O padrão AMQP completo é composto por um cliente, um roteador e um *broker*, mas esse artigo foca a troca de mensagens apenas entre cliente e *broker*. É possível encontrar implementações gratuitas e pagas [28] da especificação do protocolo.

O protocolo AMQP foi projetado para ser seguro: nenhuma mensagem pode ser revelada ou alterada por outros; confiável: há meios de garantir a entrega da mensagem; aberto e padronizado: a especificação do protocolo está disponível para todos, então é possível que diferentes implementações do AMQP conversem entre si. A versão 1.0 da especificação do AMQP define um protocolo de linha (protocolo fim-a-fim independente da aplicação) para comunicação de mensagens entre cliente e *broker* [13]. Também especifica um roteador que pode alterar a mensagem e, baseado em um critério ou conjunto de regras, decide para qual fila a mensagem deve ser encaminhada, como representado na Fig. 6.

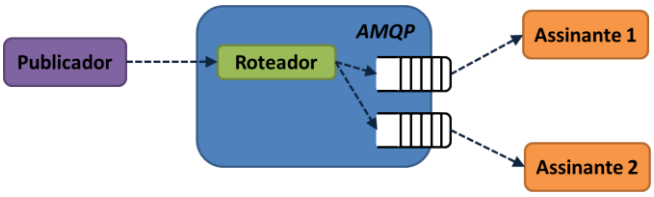

Fig. 6. Arquitetura AMQP.

A conexão é sempre estabelecida do cliente para o *broker* de mensagens. Primeiro, o cliente precisa abrir um socket TCP e as mensagens trocadas inicialmente definem as funcionalidades e limitações de cada lado. Para redes limitadas, pode ser caro trocar essa informação a cada conexão, então o AMQP tem um mecanismo para omitir algumas mensagens de negociação em conexões consecutivas. Antes que as primeiras transferências sejam feitas, um lado precisa criar uma Sessão[2] e um Enlace[3]. Cada Sessão precisa ter muitos Enlaces associados e um Enlace sempre precisa estar associado a uma Sessão. Depois de estabelecer a conexão, o cliente e o *broker* começam a troca de mensagens e o Enlace mantém informações sobre cada transferência e controla o fluxo de mensagens; se o destinatário, por alguma razão, não receber uma mensagem, ele informará à fonte que pode esperar por uma entrega futura.

Uma mensagem pode chegar ao destino usando outros nós de rede ou através do *broker* de mensagens. A mensagem tem atributos para definir o tipo de conteúdo da mensagem e como a mensagem atravessa os nós da rede. Para redes complexas, a mensagem pode informar regras de roteamento e políticas para armazenamento da mensagem em nós intermediários. O conteúdo da mensagem pode ser qualquer tipo de dado e é responsabilidade do alvo saber como interpretá-lo corretamente, mas é possível configurar um atributo MIME (*Multi-purpose Internet Mail Extension*) para instruir como o conteúdo da mensagem deve ser interpretado. A composição da mensagem é mostrada na Fig. 7. A seção marcada como "Bare Message" (mensagem crua) é o conteúdo da aplicação e é imutável através da rede. Outras partes da mensagem podem ser populadas pelo transmissor AMQP e outros nós conforme a mensagem atravessa a rede.

[2] Sessões são vias de comunicação sequencial bidirecional e provêm controle de fluxo para as transferências de mensagens.

[3] Enlace é um caminho unidirectional criado entre dois lados: uma fonte e um alvo.

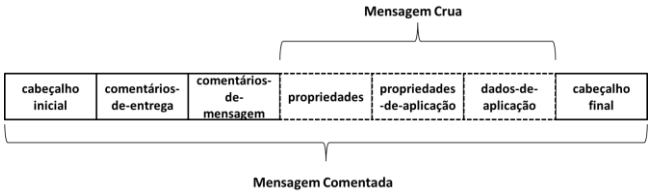

Fig. 7. Formato da mensagem AMQP.

Há papeis do Enlace podem ser definidos no momento da criação para diferentes padrões de transferência de mensagens que representam o nível de QoS desejado na comunicação. Os possíveis modos de entrega de mensagens são *pelo menos uma vez*, *no máximo uma vez* e *exatamente uma vez*, e são similares aos padrões de QoS do MQTT.

A segurança do AMQP baseada no TLS sobre TCP e autenticação simples e camada de segurança (SASL — *Simple Authentication and Security Layer*) também pode ser usada. O conteúdo pode ser criptografado em algum nível para aumentar a segurança da comunicação.

Como esse protocolo é mantido para funcionar em grandes negócios, ele também é muito completo e complexo. O uso desse protocolo para IoT é principalmente influenciado pela alta disponibilidade de implementações de clientes e *brokers* que tornam a interoperabilidade com sistemas legados possível.

III. Comparação dos Protocolos Apresentados

Uma vez que esse artigo foca a comparação dos protocolos propostos para ambientes limitados, a atenção na análise qualitativa é voltada para a definição das características do protocolo que melhor se adequam ao cenário de comparação. Como definido por Sen (2010), os principais limitantes em redes de sensores sem fio (WSNs), que são a base para cenários IoT [30], [31], são: limitantes de energia, limitação de capacidade de processamento, redes não-confiáveis, altas latências na comunicação e operação da rede sem supervisão. De acordo com as características de limitação relacionadas, esse artigo adota um cenário de referência, como mostrado na Fig. 8, composto por nós sensores "S", usando microcontroladores de 8–32 bits com taxa de clock de 16–190 MHz, 8k–64M bytes de memória RAM (DRAM, SRAM) e 64k–1M bytes de memória flash [31]. Todos os sensores são alimentados por baterias e trocam dados em uma rede não-confiável que pode ser com ou sem fio. Considera-se a transferência de dados entre os nós "S" na faixa de 15 kbps a 1 Mbps. Os dados coletados são enviados para o ambiente de computação em nuvem através de um *Gateway*, com uma taxa mínima de 50 kbps, e são disponibilizados para os usuários de alguma forma.

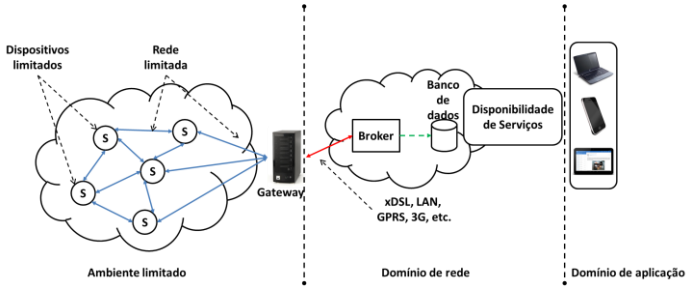

Fig. 8. Cenário de referência com ambiente limitado para aplicações IoT.

Algumas métricas devem ser definidas para popular a comparação entre os protocolos propostos. Para uma análise exaustiva, os seguintes tópicos serão considerados: Implementação (custo em tamanho); Transporte de dados (custo de transmissão); Padrões de comunicação; Confiabilidade e QoS; Escalabilidade; e Segurança.

A. *Implementação*

Em termos de implementação, o MQTT tem a especificação de protocolo mais simples [11] e, portanto, facilita o desenvolvimento do cliente. Os clientes CoAP atuam como clientes HTTP mas em modo binário, o que se torna mais simples que o HTTP, mas ainda mais complexo que o MQTT. Do ponto de vista de implementação, o AMQP tem o cliente mais complexo, o que o torna menos atraente para dispositivos limitados. Assim, encaixa-se melhor na implementação do *Gateway* (Fig. 8). O MQTT e o CoAP são mais adequados para a implementação dos nós "S" onde há limitações de energia, capacidade de processamento e memória.

B. *Transporte de dados*

Tanto o MQTT quanto o AMQP empregam uma comunicação orientada a conexão dada pelo TCP, o que custa mais que o UDP usado pelo CoAP. O uso do TCP significa mais dados trocados entre o cliente e o servidor. Se o TCP ou o UDP não são necessários, uma alternativa é escolher a abordagem do MQTT-SN sobre 6LoWPAN (IPv6 sobre redes sem fio de área pessoal de baixa potência)[4] ou mesmo do ZigBee®[5], evitando a complexidade da pilha TCP/IP completa. O CoAP também foi projetado para redes limitadas, como o 6LoWPAN, com o fim de manter o *overhead* de mensagens pequeno, assim limitando a necessidade de fragmentação que causa redução significativa na probabilidade de entrega de pacotes [12].

O formato da mensagem no MQTT e no CoAP é binário, enquanto o AMQP usa mensagens formatadas XML, mas também pode transferir dados binários. O cabeçalho MQTT tem apenas dois bytes, tornando-o uma interessante solução para redes com baixas taxas de transmissão, o que representa, no cenário de referência proposto (Fig. 8), a rede entre os nós

[4] 6LoWPAN: mecanismos de compressão de cabeçalho e encapsulamento que permitem que pacotes IPv6 sejam enviados e recebidos em redes baseadas no IEEE 802.15.4;

[5] ZigBee®: especificação de um conjunto de protocolos de comunicação de alto nível usados para criar redes de área pessoal construídas com rádios digitais pequenos e de baixa potência. O ZigBee é baseado no padrão IEEE 802.15.4;

"S".

No caso dos dados úteis, o MQTT é agnóstico e os dados podem ser transmitidos sem um tipo específico ou formato. O CoAP funciona com dados úteis binários e o AMQP com um arranjo mais complexo de dados úteis que podem definir regras e propriedades. O uso de dados úteis baseados em *string* ou XML torna necessário o uso de interpretadores mais complexos, aumentando os requisitos de hardware. Pode ser mais plausível em implementações de *Gateways* (Fig. 8), onde mais recursos como memória e capacidade computacional estão disponíveis.

Considerando uma banda limitada, o MQTT e o CoAP são mais adequados que o AMQP.

C. Padrões de comunicação

Antes de fazer a comparação, é necessário introduzir os padrões de comunicação IoT que podem ser definidos como *Telemetria*, *Consulta*, *Comando* e *Notificação*. Em *Telemetria*, a informação flui dos dispositivos para a nuvem informando mudanças de estados nos dispositivos. A Fig. 9 apresenta um exemplo de padrão de comunicação *Telemetria* para os protocolos (a) MQTT, (b) CoAP e (c) AMQP, respectivamente. De acordo com a Fig. 9 (b), o padrão *Telemetria* não é adequado para o CoAP porque a conexão precisa ser iniciada do sistema (cliente) para o dispositivo (servidor), que pode enfrentar problemas de endereçamento como *roaming* móvel ou NAT. O modelo publica/inscreve do MQTT é equivalente ao padrão *Telemetria*, facilitando seu uso. O AMQP é apropriado quando um melhor controle de fluxo e segurança são desejáveis.

Para o padrão *Consulta*, as requisições vêm dos dispositivos para a nuvem para coletar informações requeridas, como ilustrado na Fig. 10 para os protocolos (a) MQTT, (b) CoAP e (c) AMQP, respectivamente. De acordo com Fig. 10 (b) e (c), o CoAP e o AMQP têm melhor desempenho para esse padrão uma vez que são baseados no modelo requisição/resposta. Quando se usa o MQTT para o padrão *Consulta*, há necessidade de definir o tópico de resposta para a comunicação uma vez que não há um caminho de resposta construído. Isso configura uma dificuldade de implementação.

No padrão *Comandos*, comandos são enviados dos sistemas para os dispositivos para desempenhar atividades específicas. A Fig. 11 apresenta exemplos para os protocolos (a) MQTT, (b) CoAP e (c) AMQP, respectivamente. Analisando esse cenário, o AMQP é mais adequado porque tem suporte nativo a resultados e correlação. É possível usar "replyTo" e "correlation_id" para o fluxo de controle comando/resultado mesmo que o dispositivo esteja desligado. Além disso, pode-se usar o TTL (*Time to Live*) para evitar a entrega de comandos antigos. O CoAP apresenta, para esse padrão, os mesmos problemas de endereçamento detalhados em *Telemtria*. No caso do MQTT, não há suporte a caminhos de resultados nativos, o que requer a definição de um tópico de resultado. Ainda, não há TTL para comandos, sendo que comandos antigos podem ser entregues quando se usa a *flag* "retain" ou novos comandos podem ser perdidos quando não é usada.

Por fim, em *Notificação*, a informação flui dos sistemas para os dispositivos, lidando com mudanças de estados no mundo físico como mostrado na Fig. 12 para os protocolos (a) MQTT, (b) CoAP e (c) AMQP, respectivamente. Nesse padrão de comunicação, os problemas de endereçamento do CoAP, também relacionados nos padrões *Comando* e *Telemetria* estão presentes. Por outro lado, o modelo publica/assina do MQTT se adequa à arquitetura de notificação apresentando problemas apenas se um melhor controle de fluxo é necessário para uma grande quantidade de dados a taxas altas, caso em que o AMQP tem melhor desempenho.

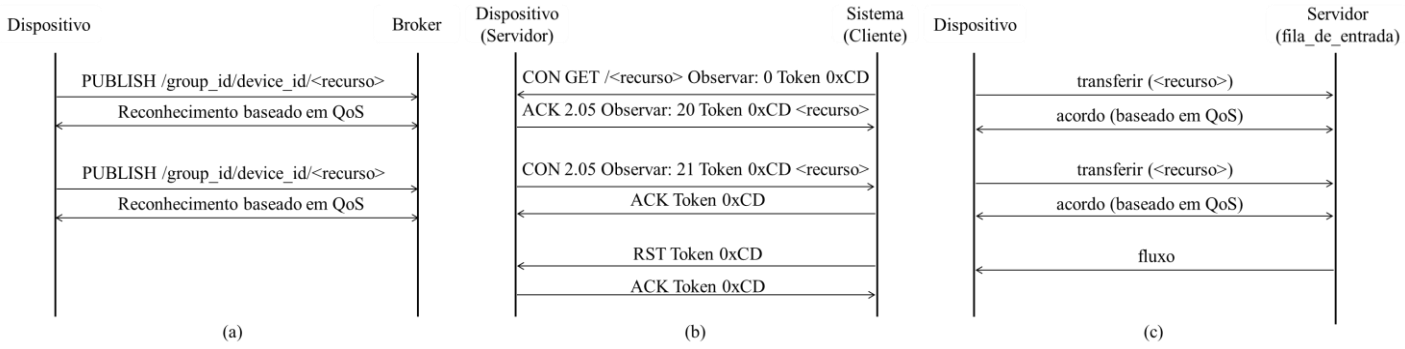

Fig. 9. Exemplo de padrão de comunicação *Telemetria* para (a) MQTT, (b) CoAP e (c) AMQP.

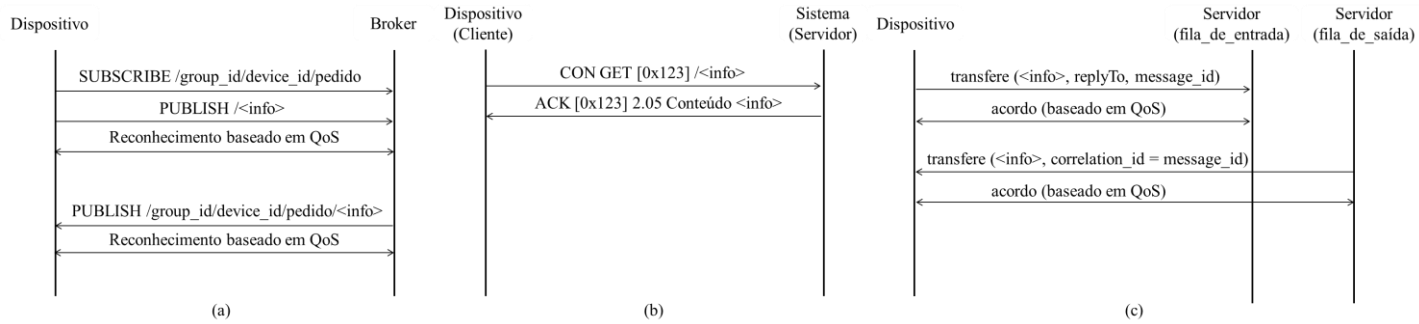

Fig. 10. Exemplo de padrão de comunicação *Consulta* para (a) MQTT, (b) CoAP e (c) AMQP.

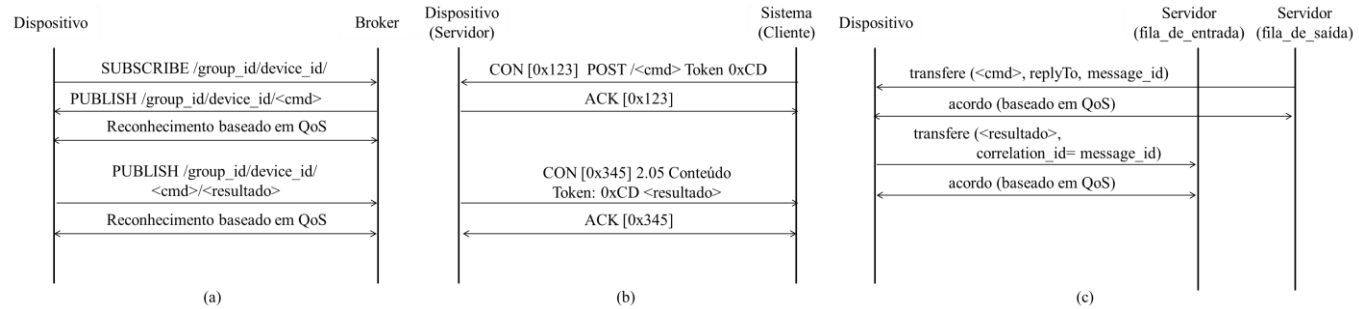

Fig. 11. Exemplo de padrão de comunicação *Comando* para (a) MQTT, (b) CoAP e (c) AMQP.

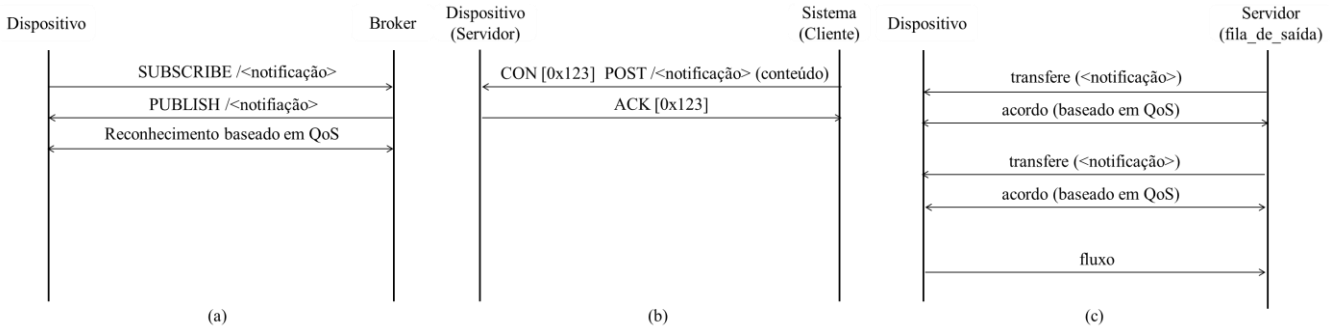

Fig. 12. Exemplo de padrão de comunicação *Notificação* para (a) MQTT, (b) CoAP e (c) AMQP.

D. Confiabilidade e QoS

Conforme apresentado na última subseção, todos os padrões de comunicação podem ter um aumento de confiabilidade usando algum nível de QoS, o que garante a entrega dos dados ou mesmo evita duplicação de pacotes. Os três protocolos apresentados têm opções de QoS que podem ser usadas dependendo do controle de fluxo de dados desejado. O que precisa ser considerado é que quanto mais alto o nível de QoS usado, menor a perda de pacotes porém maior o atraso nas comunicações [18].

E. Escalabilidade

Arquiteturas baseadas nos protocolos MQTT ou AMQP podem facilmente ser escaladas horizontalmente porque são baseadas em modelos publica/assina. A força do modelo é baseada no desacoplamento no tempo onde quem publica e quem assina não precisa transmitir ao mesmo tempo. Além disso, quem publica e quem assina não precisam saber um a respeito do outro, o que representa um desacoplamento no espaço. Eventos podem ser produzidos ou consumidos de forma assíncrona, permitindo maior escalabilidade e flexibilidade [32]. Como ambos os protocolos dependem de *brokers* para trocarem mensagens, a infraestrutura do Sistema pode ser facilmente escalada se mais banda ou poder de processamento forem necessários. Arquiteturas baseadas no CoAP também podem ser escaladas, mas de uma forma diferente uma vez que os dispositivos são considerados recursos. Porém, se a opção observar do CoAP for usada em um modelo de interação *Telemetria*, os clientes têm permissão para monitorar eventos que sejam de seu interesse por meio de uma requisição GET estendida enviada pelo nó servidor. O servidor notifica cada nó cliente que tenha uma relação de observação com o evento. Apesar de, na arquitetura geral desse modelo, o servidor agir como um *broker*, alta escalabilidade e eficiência podem ser atingidas usando *caches* e nós intermediários (*proxies*) que multiplexem o interesse de múltiplos clientes (assinantes) em um mesmo evento para uma associação única.

F. Segurança

Como visto nas descrições dos protocolos apresentadas por muitos autores [29], a segurança é um dos problemas principais a serem resolvidos nos cenários IoT. Analisando os protocolos selecionados nesse trabalho, o CoAP é baseado no DTLS e permite criptografia apenas nos dados úteis, o que provê certo aumento da segurança, mas não protege a mensagem inteira. Tanto o MQTT quanto o AMQP são baseados no TLS/SSL e usam criptografia apenas nos dados úteis. Para algumas aplicações onde a informação transferida não é sensível, o TLS/SSL pode ser muito caro computacionalmente e apenas a criptografia dos dados úteis pode ser suficiente.

A Tabela I resume a comparação entre as principais características dos protocolos M2M propostos.

Tabela I –TABELA DE COMPARAÇÃO DOS PROTOCOLOS IOT.

	MQTT	MQTT-SN	CoAP	AMQP
Protocolo de Rede	TCP/IP	Não especificado	UDP	TCP/IP
Tipo de dados úteis	Binário	Binário	Binário	*String*
Adequado para microcontroladores	Sim	Sim	Sim	Não
Segurança	SSL/TLS	Não especificado	DTLS	SASL/TLS
Escalabilidade	Simples	Simples	Complexa	Simples
Arquitetura de rede	Baseado em *broker* (publica/assina)	Baseado em *broker*, cliente/servidor, cliente/cliente	Cliente/servidor (requisição/resposta)	Baseado em *broker* (publica/assina)
Padrão de comunicação	Baseado em tópicos	Baseado em tópicos	Arquitetura REST	Distribuição baseada em nós
Opções de QoS	Sim	Sim	Sim	Sim

IV. CONCLUSÃO

O presente artigo forneceu uma comparação qualitativa entre algumas abordagens importantes para protocolos M2M aplicados a ambientes TIC limitados, mais especificamente à Internet das Coisas. Cada protocolo tem suas particularidades, que os tornam mais adequados para uma situação específica. O uso do TCP como protocolo de transporte restringe seu uso em ambientes limitados. Assim, protocolos de transporte mais leves, como o 6LoWPAN e o ZigBee®, estão sendo usados, gerando novas possibilidades para dispositivos limitados. O fluxo de mensagens de controle pode ser configurado de acordo com a necessidade usando as opções de QoS disponíveis em cada protocolo, o que por sua vez aumenta a taxa de dados necessária e o atraso. A segurança é um dos principais problemas para todas as abordagens e precisa ser revisto para futuras versões. Em termos de escalabilidade, todos os protocolos propostos são capazes de ser escalados, onde os protocolos baseados no modelo publica/assina têm implementação mais simples. Finalmente, um protocolo com mais funcionalidades pode ser usado na medida em que mais recursos se tornam disponíveis no ambiente. A escolha do microcontrolador, o custo por dado enviado, a confiabilidade da rede, a taxa de mensagens, a QoS requerida e o nível de segurança são alguns problemas a serem analisados antes de se escolher o protocolo mais apropriado. Em resumo, a escolha depende do cenário de aplicação e, de acordo com os requisitos do sistema inteiro, mais de um protocolo pode ser usado.

REFERÊNCIAS

[1] K. Ashton, “That ‘Internet of Things’ Thing,” *RFID Journal*, 1999. [Online]. Available: http://www.rfidjournal.com/articles/view?4986. [Accessed: 10-Oct-2014].

[2] W. Leister and T. Schulz, “Ideas for a Trust Indicator in the Internet of Things,” in *The First International Conference on Smart Systems, Devices and Technologies (SMART 2012)*, 2012, no. c, pp. 31–34.

[3] Y. Yu, J. Wang, and G. Zhou, “The exploration in the education of professionals in applied Internet of Things Engineering,” in *4th International Conference on Distance Learning and Education (ICDLE)*, 2010, pp. 74–77.

[4] E. Bucherer and D. Uckelmann, “Business Models for the Internet of Things,” in *Architecting the Internet of Things*, D. Uckelmann, M. Harrison, and F. Michahelles, Eds. Springer Berlin Heidelberg, 2011, pp. 253–277.

[5] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, “Key management systems for sensor networks in the context of the Internet of Things,” *Comput. Electr. Eng.*, vol. 37, no. 2, pp. 147–159, Mar. 2011.

[6] H. Ning and L. Hong, “Cyber-Physical-Social Based Security Architecture for Future Internet of Things,” *Adv. Internet Things*, vol. 02, no. 01, pp. 1–7, 2012.

[7] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks,” in *3rd*

International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), 2008, pp. 791–798.

[8] B. Krishnamachari and A. Dunkels, "Whither Sensor Networks ? Towards the Internet of Things," *Communication*, 2011.

[9] C. P. Mayer, "Security and Privacy Challenges in the Internet of Things," *Challenges*, vol. 17. 2009.

[10] M. Collina, G. Corazza, and A. Vanelli-coralli, "Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST," in *23rd Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications Introducing*, 2012, pp. 36–41.

[11] International Business Machines Corporation (IBM), "MQTT: Message Queuing Telemetry Transport, version 3.1, protocol specification." Eurotech, pp. 1–42, 2010.

[12] Z. Shelby, "RFC 7252: The Constrained Application Protocol (CoAP)." pp. 1–112, 2014.

[13] "AMQP: Advanced Message Queuing Protocol, version 1.0, AMQP working group protocol specification," vol. 2011, no. revision 0. pp. 1–112, 2011.

[14] N. O'Leary, "Paho - Open Source messaging for M2M," *Eclipse Paho's MQTT*, 2014. [Online]. Available: http://www.eclipse.org/paho/. [Accessed: 21-Dec-2014].

[15] "RabbitMQ." [Online]. Available: http://www.rabbitmq.com/. [Accessed: 19-Dec-2014].

[16] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A Low-Power CoAP for Contiki," *2011 IEEE Eighth Int. Conf. Mob. Ad-Hoc Sens. Syst.*, pp. 855–860, Oct. 2011.

[17] "Mosquitto." [Online]. Available: http://mosquitto.org/. [Accessed: 15-Dec-2014].

[18] L. Shinho, K. Hyeonwoo, H. Dong-kweon, and J. Hongtaek, "Correlation Analysis of MQTT Loss and Delay According to QoS Level," *IEEE*, pp. 714–717, 2013.

[19] M. Jamie and G. Jeremy, "Sensor Networks and Grid Middleware for Laboratory Monitoring."

[20] P. K. Tiwari, S. Parthasarathy, A. N. Chatterjee, and N. Krishna, "Integrated Wireless Sensor Network for Large Scale," *IEEE*, pp. 1849 – 1854, 2013.

[21] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Comput. Networks*, vol. 67, pp. 104–122, Jul. 2014.

[22] M. Linh, "MQTT-SN: MQTT For Sensor Networks, version 1.2, protocol specification." pp. 1–27, 2013.

[23] M. Brachmann, O. Garcia-morchon, and M. Kirsche, "Security for practical coap applications: Issues and solution approaches," *2011 10th GI/ITG KuVS Fachgespraech Sensornetze (FGSN 2011)*, no. Fgsn, pp. 1–4, 2011.

[24] T. Levä, O. Mazhelis, and H. Suomi, "Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications," *Decis. Support Syst.*, vol. 63, pp. 23–38, Jul. 2014.

[25] B. C. Villaverde, D. Pesch, R. De Paz Alberola, S. Fedor, and M. Boubekeur, "Constrained Application Protocol for Low Power Embedded Networks: A Survey," *2012 Sixth Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput.*, pp. 702–707, Jul. 2012.

[26] C. Bormann, A. P. Castellani, Z. Shelby, U. Bremen, and Z. S. Sensinode, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Comput.*, vol. 16, no. 2, pp. 62–67, 2012.

[27] S. Vinoski, "Advanced Message Queuing Protocol," *IEEE Internet Comput.*, vol. 10, no. 6, pp. 87–89, 2006.

[28] "Apache Qpid." [Online]. Available: http://qpid.apache.org/. [Accessed: 05-Jan-2015].

[29] J. Sen, "A Survey on Wireless Sensor Network Security," *Comput. Networks*, vol. 52, no. 12, p. 24, 2010.

[30] H. Petersen, E. Baccelli, and W. Matthias, "Interoperable Services on Constrained Devices in the Internet of Things," pp. 1–4.

[31] J. H. Kong, L.-M. Ang, and K. P. Seng, "A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments," *J. Netw. Comput. Appl.*, pp. 1–36, Oct. 2014.

[32] E. G. Davis, A. Calveras, and I. Demirkol, "Improving packet delivery performance of publish/subscribe protocols in wireless sensor networks.," *Sensors (Basel).*, vol. 13, no. 1, pp. 648–80, Jan. 2013.