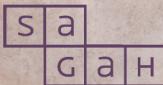


ARQUITETURA E INFRAESTRUTURA DE IOT

Luis Gustavo Maschietto



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

Protocolo MQTT

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar os objetivos do protocolo MQTT.
- Reconhecer as características do protocolo MQTT.
- Descrever a estrutura do protocolo MQTT.

Introdução

Existem diversos protocolos de comunicação presentes na Internet das Coisas (IoT, do inglês *Internet of Things*) cujo uso é fundamental, pois possibilita a transmissão de dados entre objetos em uma rede. Entre os protocolos existentes, o Message Queue Telemetry Transport (MQTT) apresenta-se como o mais importante, principalmente por permitir um ótimo desempenho de transmissão em situações de recursos limitados.

Neste capítulo, você vai aprender a identificar os objetivos do protocolo MQTT, seus criadores e as diferenças existentes em relação a outros protocolos usados em IoT. Além disso, saberá reconhecer as características essenciais do protocolo MQTT e compreenderá como sua estrutura é descrita.

1 Objetivos do protocolo MQTT

A evolução de estruturas computacionais permitiu que novos recursos de comunicação pudessem ser viáveis, principalmente em relação a dispositivos de pequeno porte como celulares, *tablets*, *smartwatches*, óculos inteligentes, entre outros dispositivos. Com o avanço das tecnologias de comunicação, microeletrônica e de computação, o conceito de IoT expandiu-se rapidamente e permitiu o desenvolvimento de aplicações em diferentes ramos de atividade humana. Assim, seu uso ganhou interesse social expressivo com as vendas de diversos produtos inteligentes. A popularização de sistemas embarcados e a utilização de armazenamento em nuvem, *hardware* e sensores fizeram com que

várias empresas iniciassem a fabricação de seus dispositivos de IoT. Gigantes como Samsung, ARM, Intel, Microsoft, IBM, Google e Apple contam com equipes empenhadas na criação de soluções para IoT. Em relação à criação de protocolos, a ideia segue a mesma: várias iniciativas de implementações, com foco na eficiência e qualidade das comunicações entre dispositivos, estão sendo criadas a todo momento (O PROTOCOLO..., 2018).

Segundo Torres, Rocha e Souza (2016), por estar em rápida evolução, a IoT possibilita que diversos protocolos sejam apresentados com o objetivo de se tornarem candidatos a um padrão de mercado. Em relação a esses protocolos, é possível citar o Constrained Application Protocol (CoAP), Extensible Messaging and Presence Protocol (XMPP), Simple Network Management Protocol (SNMP), Web Application Messaging Protocol (WAMP) e o MQTT. Em todos esses protocolos, é utilizada a comunicação máquina a máquina (*machine-to-machine* — M2M).

Para Martins e Zem (2015), as aplicações do tipo M2M apresentam grande potencial para se tornarem uma tendência no desenvolvimento de *software*, principalmente por englobarem diversas áreas da economia e por estarem presentes em ambientes domésticos com soluções automatizadas de interação. As pesquisas relacionadas à comunicação M2M ganham importância porque:

- a tecnologia M2M é um assunto em expansão;
- a cada dia a computação está ficando mais ubíqua e essa imensa disponibilidade de conectividade é um cenário ideal para a exploração de tecnologias M2M;
- a computação ubíqua torna o ambiente propício para investimentos em tecnologias que coloquem não apenas supercomputadores na rede, mas também dispositivos restritos, de pequeno porte, com baixo consumo de energia e baixa capacidade computacional (como memória e processamento);
- com o advento do Internet Protocol Versão 6 (IPV6), a capacidade de endereçamento de nós na rede abre espaço para um número imenso de dispositivos conectados.

Para Martins e Zem (2015), a M2M é considerada uma tecnologia precursora da IoT, pois possui soluções baseadas em comunicações ponto a ponto que utilizam dispositivos embarcados com o objetivo de reduzir custos de gerenciamento de recursos (diagnósticos remotos, suporte técnico ou atualizações), ao passo que as soluções de IoT envolvem acessos muito mais amplos, como ocorre na acomodação de sensores passivos e entre outros objetos, para permitir

uma melhoria não apenas na produção e prestação de serviço das empresas, mas também no modelo de negócio.

Nos anos 1990, a IBM, com a ajuda dos desenvolvedores Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech), disponibilizou o protocolo MQTT com a intenção de padronizar as comunicações realizadas via IoT. A tecnologia MQTT, padrão da IoT, foi desenvolvida com base na pilha TCP/IP, que é um protocolo de rede subjacente da internet (YUAN, 2017).

O objetivo central do protocolo MQTT é permitir a comunicação entre objetos da IoT e definir o papel de cada objeto a partir de um modelo de operação. Entre as operações fundamentais está a definição do papel de cada dispositivo, a ordem de envio de mensagens e o formato das mensagens. O MQTT foi criado para ser empregado em dispositivos com baixa largura de banda, capacidade computacional reduzida e sem garantia de conectividade (MENEZES *et al.*, 2017). Inicialmente, segundo Yum (2017), sua aplicação era voltada a sensores em *pipelines*, como satélites e petróleo.

Em 2014, com a publicação do RFC 7252, o CoAP se tornou o protocolo-padrão para a troca de informações entre dispositivos com restrições computacionais, restrições de energia e em redes com baixa largura de banda (MARTINS; ZEM, 2015). Mais recentemente, porém, a tecnologia que apresenta maior adesão pelas empresas do ramo de IoT é o MQTT. A Amazon, por exemplo, adotou MQTT, WebSocket e Hypertext Transfer Protocol (HTTP) como protocolos-padrão em sua plataforma AWS IoT. Já o Facebook, mesmo fora do ramo de IoT, adotou em suas comunicações um sistema de *instant messaging* que utiliza o protocolo MQTT. Dessa forma, fica claro que o protocolo MQTT é bastante relevante no cenário atual (TORRES; ROCHA; SOUZA, 2016).

De acordo com Vanzella *et al.* (2018), quando comparado ao MQTT, o CoAP revela-se mais leve, por rodar sobre User Datagram Protocol (UDP). Porém, como o UDP não possui mecanismos de retransmissão, a perda de pacotes é mais provável ao se utilizar CoAP. Em termos de *performance*, atrasos em transmissão e utilização de banda, resultados experimentais revelam que mensagens MQTT apresentam menor atraso que as mensagens CoAP quando as taxas de perda de pacotes são baixas, embora apresentem maior atraso quando as taxas de perda são maiores.

O MQTT pode ser aplicado em diferentes soluções de IoT, como no controle de equipamentos elétricos nas indústrias, controle de equipamentos de segurança, controle de equipamentos eólicos e fotovoltaicos, entre outras aplicações (CONCEIÇÃO; COSTA, 2019). Como recurso para troca de mensagens, o protocolo MQTT apresenta um servidor de aplicação chamado Broker. A função

do Broker é servir como um sistema intermediário nas trocas de mensagens. A Figura 1 ilustra mensagens sendo publicadas em um servidor MQTT Broker por diversos dispositivos de diferentes finalidades ao redor do mundo.

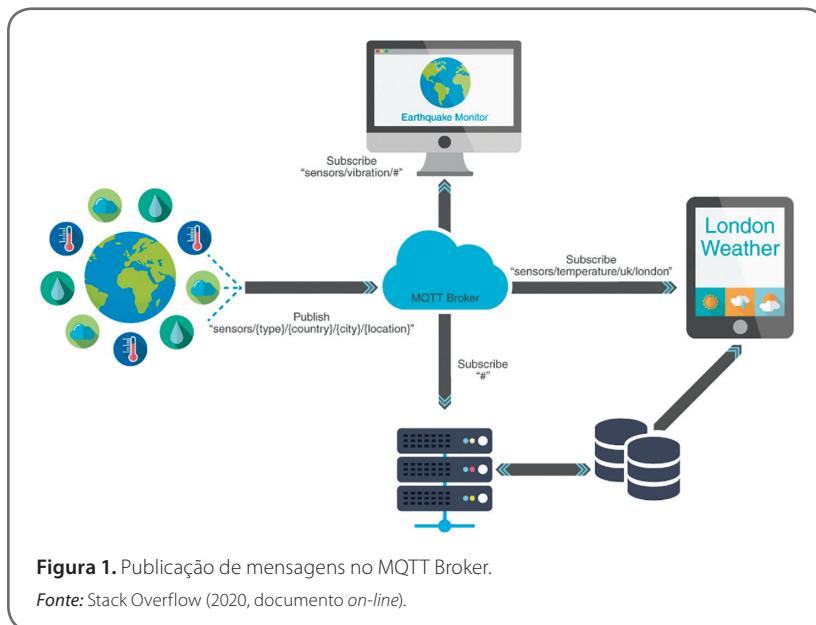


Figura 1. Publicação de mensagens no MQTT Broker.

Fonte: Stack Overflow (2020, documento *on-line*).

Esse alcance no envio de dados só é possível graças aos protocolos de comunicação em rede e dos protocolos envolvidos nos dispositivos de IoT. Nesse exemplo, é possível notar que mensagens publicadas pelos dispositivos sob protocolo MQTT podem ser acessadas via Broker a partir de aplicações *web* e aplicações *mobile*. Dessa forma, de acordo com a Figura 2, todo conteúdo compartilhado entre dispositivos de IoT são armazenados em sistemas gerenciadores de bases de dados para uma posterior análise a partir de gráficos gerados pelas aplicações desenvolvidas. Portanto, é possível afirmar que o protocolo MQTT é fundamental para a subsistência da IoT, principalmente em relação aos dados compartilhados por dispositivos sem fio através da rede de computadores.

Na próxima seção, serão apresentadas as principais características do protocolo MQTT e os principais parâmetros existentes na troca de informações entre dispositivos.

2 Características do protocolo MQTT

Quando comparado a outras tecnologias, o protocolo MQTT é relativamente simples, pois apresenta como características implementações menos complexas, além de garantir maior qualidade em seus serviços (com confiabilidade na entrega das mensagens com o uso de recurso Quality of Service — QoS). Em relação à segurança, apesar de não ser projetado para esse fim, como no caso do Secure Sockets Layer (SSL) ou do Transport Layer Security (TLS), o MQTT apresenta recursos de autenticação por credenciais (controle por senha e usuário). Por apresentar essas características, o protocolo possui maior aceitação em implementações de sistemas embarcados, principalmente por não exigir alto consumo de bateria e por operar satisfatoriamente em dispositivos com baixo poder de processamento. Apesar de estar na camada OSI junto com o HTTP, o MQTT apresenta *payload* menor. Isso significa que ele consegue se manter estável mesmo com baixa qualidade de conexão. Outra vantagem em comparação ao HTTP é que o MQTT possibilita uma comunicação entre dispositivos na ordem 1 para N, ou seja, a comunicação realizada por um dispositivo pode envolver vários outros (N) (O PROTOCOLO..., 2018).

Como já explicado, a função do MQTT Broker é atuar como intermediário no intercâmbio de mensagens entre dispositivos na rede (mais detalhes na próxima seção do capítulo). Para um cliente se conectar via Broker (responsável por receber as mensagens enviadas), inicialmente o cliente envia para o Broker a instrução CONNECT a partir de mensagem contendo diversos parâmetros de execução. Os nomes dos diferentes parâmetros e suas características são os seguintes (CONCEIÇÃO; COSTA, 2019):

- **KeepAlive** — representa o tempo disponível para que o cliente possa permanecer no Broker e manter sua conexão ativa;
- **lastWillMessage** — representa a uma mensagem de último desejo;
- **lastWillQos** — o QoS da mensagem de último desejo;
- **lastWillTopic** — mensagem de último desejo em um tópico no momento de encerramento da conexão;
- **password** — credenciais de autorização e autenticação do Broker;
- **username** — credenciais de autorização e autenticação do Broker;
- **cleanSession** — verifica se a conexão armazenou as mensagens que, por ventura, possam ter sido perdidas e todas as assinaturas no Broker (conexão persistente).

A Figura 2 exibe um exemplo do uso dos parâmetros MQTT para uma conexão realizada pelo cliente via Broker em que será subscrita uma mensagem. É possível verificar na linha 6 a criação da variável `keep_alive_broker`, que controlará o período no qual o cliente poderá manter uma conexão ativa no servidor Broker. O método `on_connect`, na linha 10, é responsável por iniciar a conexão com o Broker e inscrever a mensagem (a variável `topico_subscribe` é passada ao método `subscribe` na linha 13).

```

1 import paho.mqtt.client as mqtt
2
3 #CONFIGURAÇÃO DO MQTT
4 Broker = "iot.eclipse.org" #broker publico utilizado.
5 porta_broker = 1883 #porta utilizada para comunicacao com o broker MQTT
6 keep_alive_broker = 60 #TEMPO CONFIGURADO EM SEGUNDOS do keep-alive
7 topico_subscribe = "MQTTRaspPiINCB" #topico MQTT que o programa ira "ouvir" (fazer subscribe)
8
9 #Callback - conexao ao broker
10 def on_connect(client, userdata, flags, rc):
11     print("[STATUS] Conectado ao Broker.")
12     #faz subscribe automatico no topico
13     client.subscribe(topico_subscribe)

```

Figura 2. Trecho de código com conexão ao Broker.

Fonte: Bertoleti ([2020?], documento *on-line*).

Já na Figura 3, é possível perceber na linha 30 o método de conexão (`connect`) responsável pelo envio dos parâmetros (Broker, `porta_broker` e o `keep_alive_broker`) necessários para a conexão ao Broker do MQTT.

```

16 def on_message(client, userdata, msg):
17     MensagemRecebida = str(msg.payload)
18     print("[MSG RECEBIDA] Topico: "+msg.topic+" / Mensagem: "+MensagemRecebida)
19 #programa principal:
20 try:
21     print("[STATUS] Inicializando MQTT...")
22     #inicializa MQTT:
23
24     #cria client MQTT e define funcoes de callback de conexao (client.on_connect)
25     #e recepcao de dados recebidos (client.on_message)
26     client = mqtt.Client()
27     client.on_connect = on_connect
28     client.on_message = on_message
29     #faz a conexao ao broker MQTT
30     client.connect(Broker, porta_broker, keep_alive_broker)
31     #mantem o MQTT funcionando por tempo indeterminado, sendo que todas as
32     client.loop_forever()

```

Figura 3. Trecho de código com passagem de parâmetro `KeepAlive`.

Fonte: Bertoleti ([2020?], documento *on-line*).

Como forma de resposta do comando CONNECT, o cliente recebe um CONNACK, em que existem os seguintes parâmetros (CONCEIÇÃO; COSTA, 2019):

- sessionPresent — responsável por indicar se já existe uma conexão com sessão persistente (tópicos assinados com possibilidade de receber mensagens que estão ausentes);
- returnCode — retorna valores de *status*; o número 0 indica sucesso e a presença de valores diferentes indica possíveis falhas.

Na Figura 4, a linha 2 apresenta o método `connack_string`, que tem por objetivo retornar uma `String` contendo informações de *status* de retorno da mensagem. É importante ressaltar que a variável `rc`, passada como parâmetro do método `connack_string`, indica se houve a conexão ou não com o Broker. Caso ocorra a conexão, o retorno da mensagem é apresentado ao cliente.

```
1 def on_connect(client, userdata, flags, rc):
2     print("Connection returned result: "+connack_string(rc))
3
4 mqtte.on_connect = on_connect
```

Figura 4. Trecho de código que indica o retorno da mensagem CONNACK.

Fonte: Eclipse Paho ([2020?], documento *on-line*).

Em geral, o protocolo MQTT é caracterizado por conceitos básicos com o objetivo de garantir a entrega das mensagens mantendo sua máxima leveza. O protocolo apresenta as seguintes características gerais (MOTA, 2017).

- **Modelo publish/subscribe:** é baseado no princípio de publicar mensagens e assinar tópicos.
- **Tópicos e assinaturas:** as mensagens no MQTT são publicadas para tópicos, que podem ser entendidos como áreas de interesse. Os tópicos costumam ser formados por palavras separadas por uma barra (/) e que se assemelham a caminhos.
- **QoS:** o MQTT apresenta três níveis de qualidade de serviço, cada qual indicando o esforço do servidor para garantir a entrega da mensagem. São estabelecidos os seguintes níveis de QoS (O PROTOCOLO..., 2018):

- QoS 1 (no máximo uma vez, `at most once`) — sistema parecido com o UDP, ou seja, não existe a confirmação da entrega de mensagem e o cliente não precisa ficar com a mensagem guardada em retransmissões futuras.
- QoS 2 (pelo menos uma vez, `at least once`) — checagem da entrega de mensagem. Uma cópia da informação enviada é mantida no receptor em caso de *time out*. Quando a mensagem chega ao destinatário, é enviada uma mensagem ao emissor de modo a confirmar o recebimento.
- QoS 3 (exatamente uma vez, `exactly once`) — permite que a mensagem seja entregue uma vez a partir do envio de confirmação de recebimento e o *status* de retorno da verificação de efetividade dessa confirmação, ou seja, confirmação nos dois sentidos do tráfego.
- **Mensagens retidas:** o servidor mantém as mensagens mesmo após o envio para os assinantes. Se uma nova assinatura é submetida para o mesmo tópico, qualquer mensagem retida é enviada ao novo cliente assinante.
- **Sessões limpas e conexões duráveis:** garante que as assinaturas sejam removidas após o usuário desconectar do servidor via parâmetro `clean session`. Caso a resposta do parâmetro seja falsa, a conexão é tratada como durável, e as assinaturas permanecem após a desconexão.

O protocolo MQTT, assim como o HTTP, pode ser utilizado com Brokers (serviço de comunicação) em lugares onde exista conexão com a internet. O Broker pode ser disponibilizado para os usuários a partir de um serviço hospedado de maneira remota por um servidor na internet. Em relação à infraestrutura de comunicação, o MQTT pode ser implementado sobre qualquer tecnologia. Dessa forma, podem ser criados dispositivos capazes de se comunicar via cabos de rede, comunicação celular, redes sem fio, Bluetooth, entre outros (CONCEIÇÃO; COSTA, 2019).

No Quadro 1, são destaca as principais diferenças entre o protocolo MQTT e o HTTP (MOTA, 2017).

Quadro 1. Principais diferenças entre MQTT e HTPP

Aspecto	MQTT	HTTP
Projeto	Centrado em dados	Centrado em documentos
Padrão	<i>Publish/subscribe</i>	<i>Request/response</i>
Complexidade	Simples	Mais complexo
Tamanho da mensagem	Pequeno, com cabeçalho compactado	Grande
Níveis de serviço	3 níveis de QoS	Único nível de serviço
Distribuição dos dados	Suporta 1 para 0, 1 para 1, e 1 para N	Apenas 1 para 1

Fonte: Adaptado de Mota (2017).

Após estudarmos as características do protocolo MQTT, na próxima seção examinaremos a estrutura que compõe tal protocolo e as principais camadas de rede utilizadas para a comunicação entre os dispositivos de IoT.



Saiba mais

De acordo com Oliveira (2017), como o protocolo MQTT foi desenvolvido para ser M2M, foi concebido sem qualquer tipo de interface-padrão, forma de visualização ou análise. Sendo assim, fica cabendo à aplicação de apresentação, seja para *smartphone*, *web* ou *desktop*, decidir como expor essas informações.

3 Estrutura do protocolo MQTT

Como vimos na seção anterior, o protocolo MQTT foi desenvolvido para permitir que sua infraestrutura interaja com os protocolos TCP/IP em projetos que possuam menor taxa de transferência na rede e com *hardware* leve e simples

(O PROTOCOLO..., 2018). O Protocol Data Unit (PDU) do protocolo MQTT é encapsulado pelo protocolo Transmission Control Protocol (TCP), ou seja, os dados e o cabeçalho do MQTT são enviados na área de dados do TCP. O MQTT é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes (MENEZES *et al.*, 2017). Segundo Conceição e Costa (2019), a distribuição dos protocolos nas camadas pode ser esquematizada da seguinte forma:

- camada de aplicação — MQTT;
- camada de transporte — TCP;
- camada de rede — IPv4/IPv6;
- intrarrede — Ethernet, WiFi, etc.

Assim, o protocolo MQTT é responsável por definir o modelo de operação entre os equipamentos, especificando seus papéis, a forma como será disposta as mensagens e a ordem entre elas. As funcionalidades da rede são mantidas pelas camadas TCP, pelo protocolo IP e por outras tecnologias de comunicação na camada inferior dessa arquitetura (como WiFi, Ethernet, entre outras). Com o uso do protocolo IP, os dados podem ser transmitidos pela internet.

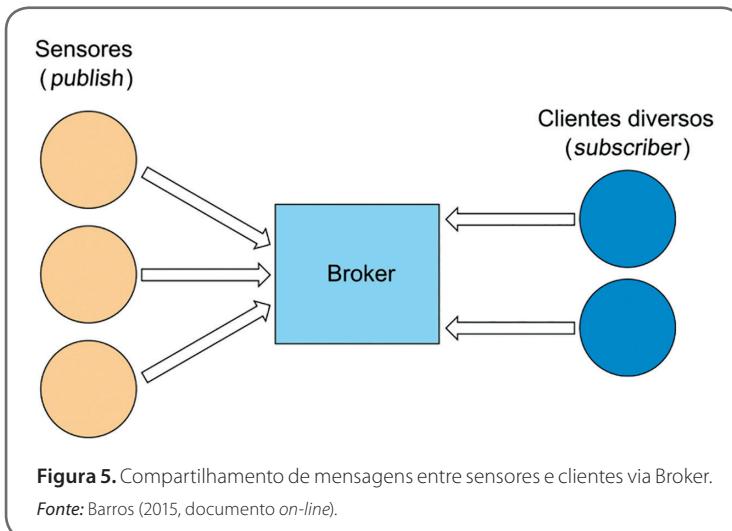


Exemplo

Aplicações que usam o protocolo MQTT incluem registros do tipo *datalog* (registro de atividades) e recursos de desenvolvimento simples para permitir, por exemplo, a programação de portas de entrada e saída (conhecidas como General Purpose Input/Output — GPIO) ou o controle do nível de velocidade de um motor com o uso de modulação por largura de pulso (*pulse-width modulation* — PWM). Uma desvantagem é que em aplicações de controle em tempo real, a partir do uso de interface de usuário, o protocolo MQTT não consegue garantir acionamentos imediatos, o que pode ocasionar uma lentidão de segundos entre uma ação e outra.

Na Figura 5, é possível verificar o padrão utilizado para o compartilhamento de mensagens no protocolo MQTT denominado publicador/subscritor, ou *publish/subscriber*. Sua função é subscrever a informação enviada para um dispositivo da rede a partir de outro elemento capaz de controlar as publicações e subscições (elemento Broker). Portanto, o elemento Broker tem a finalidade

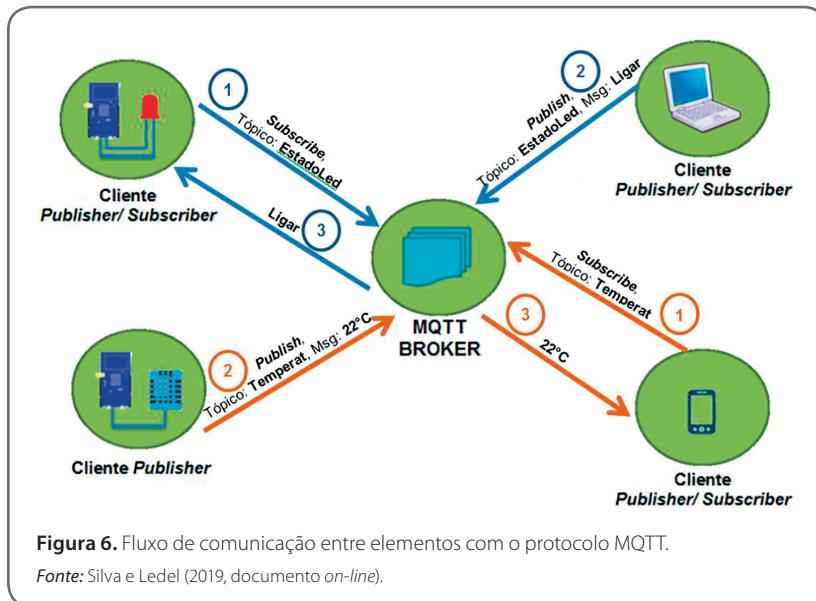
de intermediar o processo de comunicação, seja de publicação ou subscrição. O Broker permite um desacoplamento entre as partes comunicantes diferente do que ocorre sob os modelos de comunicação cliente/servidor (BARROS, 2015).



Saiba mais

No Broker, há duas formas de implementar recursos com o objetivo de aumentar a segurança na rede MQTT. A primeira é com o uso do SSL (Secure Sockets Layer), que é uma camada de transporte de segurança que permite criptografar os dados trafegados. A outra se dá mediante a utilização de autorização e autenticação com nome, senha e com uso opcional de certificados X.509. Para informações detalhadas sobre certificados digitais, procure por “Certificados do cliente X.509” no Guia do desenvolvedor AWS IoT (AWS, 2020).

Na Figura 6, é possível verificar o fluxo de comunicação que se estabelece entre o elemento que publica informações (neste caso, sensores que medem a temperatura) e aquele que assina para receber tais informações (neste caso, dispositivos [clientes] representados por aplicações que rodam na nuvem ou por pequenos computadores com baixo poder de processamento chamados de Raspberry).



Na prática, os dados no MQTT são repassados a partir de tópicos e sua estrutura de comandos é semelhante à Uniform Resource Identifier (URI), pois utiliza uma sequência de caracteres que contém níveis separados por barras (/), identificando um recurso específico, como, por exemplo, o “<http://>”. Elementos presentes em uma rede podem enviar alguns tópicos via broker, enquanto subscritores escolhem os tópicos que desejam subscrever. A sessão, estabelecida a partir da conexão entre o cliente e um Broker, é iniciada quando um cliente MQTT conecta-se em um canal de telemetria. Em relação ao processo responsável pela conexão do cliente, é possível escolher se será iniciada uma sessão limpa (nova sessão) ou uma sessão já existente. Já uma assinatura é definida pelo vínculo do cliente a um tópico (O PROTOCOLO..., 2018).



Referências

AWS. *Certificados do cliente X.509*. [S. I.]: AWS, 2020. Disponível em: https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/x509-client-certs.html. Acesso em: 11 ago. 2020.

BARROS, M. MQTT — protocolos para IoT. In: EMBARCADOS. [S. I.: s. n.], 2015. Disponível em: <https://www.embarcados.com.br/mqtt-protocolos-para-iot/>. Acesso em: 11 ago. 2020.

BERTOLETI, P. *Primeiro programa Phyton com MQTT na Raspberry Pi Introdução*. [S. I.: s. n., 2020?]. Disponível em: <https://www.newtoncbraga.com.br/index.php/microcontrolador/54-dicas-de-pic/16748-primeiro-programa-phyton-com-mqtt-na-raspberry-pi-introducao>. Acesso em: 11 ago. 2020.

CONCEIÇÃO, W. N. E.; COSTA, R. M. de R. Análise do Protocolo MQTT para Comunicação IoT através de um Cenário de Comunicação. *Caderno de Estudos em Sistemas de Informação*, v. 5, n. 2, 2019. Disponível em: <https://seer.cesjf.br/index.php/cesi/article/view/1688/1231>. Acesso em: 11 ago. 2020.

ECLIPSE PAHO. *Python client: documentation*. [S. I.: s. n., 2020?]. Disponível em: <https://www.eclipse.org/paho/clients/python/docs/>. Acesso em: 11 ago. 2020.

MARTINS, I. R.; ZEM, J. L. Estudo dos protocolos de comunicação MQTT e CoAP para aplicações machine-to-machine e Internet das coisas. *Revista Tecnológica da Fatec Americana*, v. 3, n. 1, p. 64–87, 2015. Disponível em: <https://fatecbr.websiteseugro.com/revista/index.php/RTecFatecAM/article/view/41/50>. Acesso em: 11 ago. 2020.

MENEZES, A. R. de F. et al. Internet das coisas e os principais protocolos. *Revista Expressão Científica*, v. 2 n. 2, p. 43–56, 2017. Disponível em: <https://repositorio.ifs.edu.br/biblioteca/handle/123456789/779>. Acesso em: 11 ago. 2020.

MOTA, L. da C. *Uma análise comparativa dos protocolos SNMP, Zabbix e MQTT, no contexto de aplicações de internet das coisas*. 2017. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Sergipe, São Cristovão, 2017. Disponível em: https://ri.ufs.br/bitstream/riufs/10776/2/LEVI_COSTA_MOTA.pdf. Acesso em: 11 ago. 2020

O PROTOCOLO MQTT: leve e simples: perfeito para IoT e sistemas embarcados. [S. I.: s. n.], 2018. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/mqtt/>. Acesso em: 11 ago. 2020.

OLIVEIRA, S. *Internet das coisas com ESP8266, Arduino e Raspberry Pi*. São Paulo: Novatec, 2017.

SILVA, M. R. B.; LEDEL, L. C. *RaspBene IoT*: uma plataforma de hardware e software aplicada ao monitoramento e controle de ambientes físicos. [S. l.: s. n.], 2019. Disponível em: https://hto.ifsp.edu.br/portal/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2019/TCC_MarcioRicardoBenetassoDaSilva.pdf. Acesso em: 11 ago. 2020.

STACK OVERFLOW, P. *MQTT-Broker meets WebServer with Database*. [S. l.: s. n.], 2020. Disponível em: <https://stackoverflow.com/questions/44924983/mqtt-broker-meets-webserver-with-database?rq=1>. Acesso em: 11 ago. 2020.

TORRES, A. B. B.; ROCHA, A. R.; SOUZA, J. N. Análise de desempenho de brokers MQTT em sistema de baixo custo. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 36.; WORKSHOP EM DESEMPENHO DE SISTEMAS COMPUTACIONAIS E DE COMUNICAÇÃO, 15., 2016. *Anais eletrônicos...* [S. l.: SBC], 2016. p. 47–58. Disponível em: <https://sol.sbc.org.br/index.php/wperformance/article/download/9727/9623>. Acesso em: 11 ago. 2020

VANZELLA, A. et al. *Sistema de detecção de queda e monitoramento da frequência cardíaca utilizando ESP8266 e Protocolo MQTT*. 2018. Dissertação (Mestrado em Engenharia Elétrica e Computação) — Universidade Estadual do Oeste do Paraná, Foz do Iguaçu, 2018. Disponível em: http://tede.unioeste.br/bitstream/tede/4119/5/Alanna_Vanzella_2018.pdf. Acesso em: 11 ago. 2020

YUAN, M. Conhecendo o MQT. In: IBM Developer. [S. l.]: IBM, 2017. Disponível em: <https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot/>. Acesso em: 11 ago. 2020.

Leitura recomendada

SINCLAIR, B. *IoT: como usar a "Internet Das Coisas" para alavancar seus negócios*. São Paulo: Autêntica, 2018.



Fique atento

Os *links* para sites da web fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS