UNIVERSIDADE TÉCNOLÓGICA FEDERAL DO PARANÁ

DANIEL AUGUSTO MÜLLER

PROTOCOLO MQTT: OBJETIVO, ESTRUTURA, FUNCIONAMENTO, INSTALAÇÃO E CONFIGURAÇÃO

SUMÁRIO

CAPÍTULO 1: INTRODUÇÃO

A Internet das Coisas (do inglês *Internet of Things* (IoT)) surgiu do avanço tecnológico de várias áreas como sistemas embarcados, microeletrônica, comunicação e sensoriamento (Santos, 2016). Brevemente, a IoT proporciona que objetos com capacidade computacional e de comunicação, se conectem entre si e à internet, viabilizando controle remoto, *wireless* por exemplo.

As pesquisas em IoT cresceram exponencialmente nos últimos anos. Assuntos como Casas, e até mesmo Cidades, inteligentes já não são mais novidades no mundo de hoje. Já existem inúmeros objetos como fogões, geladeiras, cortinas, lâmpadas e portões de garagem, vendidos em escala comercial com funções controladas remotamente via internet. Porém, com tantas conexões diferentes, é imprescindível a padronização dessas tecnologias de comunicação.

Os protocolos são, resumidamente, padrões criados para gerenciar a transferência de dados em uma rede que conecta dois ou mais computadores de forma eficiente considerando as características e limitações impostas pelo ambiente (Martins, 2015). No caso das comunicações, dois protocolos se destacam mundialmente, o MQTT e o CoAP.

CAPÍTULO 2: PROTOCOLOS

2.1 PROTOCOLO CoAP

O CoAP (do inglês *Constrained Application Protocol*) teve sua primeira versão publicada em 2010 pelo IETF (do inglês *Internet Engineering Task Force*), e foi desenvolvido para redes restritas, com pequena quantidade de memória *RAM* e em que a taxa de perda de pacotes é alta, segundo Martins 2015. O protocolo foi projetado para aplicações M2M (do inglês *Machine to Machine*) com, por exemplo, *smart energy e* automação residencial.

Esse protocolo utiliza a interação de requisição/resposta dos dispositivos finais, por isso utiliza quatro tipos de mensagem: *Confirmable* (CON), *Non-Confirmable* (NON), *Acknowledgment* (ACK) e *Reset*. Em uma mensagem NON, não é necessário a confirmação de recebimento no destino. Já em uma CON é necessária essa confirmação. Quando a mensagem CON é recebida sem perdas de pacote o dispositivo envia uma ACK indicando que recebeu a mensagem ou *Reset* quando não foi devidamente processada (Martins 2015).

2.2 PROTOCOLO MQTT

O MQTT (do inglês *Messaging Queue Telemetry Transport*), criado em 1999, foi desenvolvido especificamente para a comunicação M2M em dispositivos restritos e redes inseguras, com baixa largura de banda e alta latência (Martins, 2015). Segundo o site do projeto, sua meta é tentar garantir a entrega da mensagem com o mínimo de recursos possíveis.

Funciona com os protocolos TCP/IP e no modelo publicação/subscrição. No MQTT, clientes enviam e recebem mensagens subscritas em tópicos, podendo ser de vários formatos como binário, data, texto, XML ou JSON. Já o servidor ou *broker* gerencia esses tópicos e transmite para os outros clientes, possibilitando que os clientes se comuniquem entre si sem que saibam quem foi o publicador da mensagem (Atmoko, 2017).

Segundo Atmoko (2017), existem quatro comandos principais: 1. *PUBLISH*, que publica a mensagem, 2. *SUBSCRIBE*, mensagem usada por clientes para especificar tópicos, 3. *UNSUBSCRIBE*, mensagem usada por clientes para remover a inscrição de tópicos, e 4. *CONNECT*, onde o cliente requisita conexão ao servidor.

CAPÍTULO 3: FUNCIONAMENTO DO PROTOCOLO MQTT

Cada mensagem MQTT tem um cabeçalho fixo de 2 *bytes*. O primeiro byte contém o tipo do pacote de controle e *flags* específicas para cada tipo de pacote de controle, e o segundo byte contém a largura restante, como pode ser visto na Figura 1.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

Figura 1 - Cabeçalho fixo de uma mensagem MQTT 5

Fonte: EBM e Eurotech (2019)

O tipo de mensagem, informado no primeiro *byte* corresponde dos *bits* 7 ao 4 e pode assumir vários valores e funções diferentes, como pode ser visto na Tabela 1.

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

Tabela 1 –Tipos de mensagem MQTT 5

Fonte: IBM e Eurotech (2019)

Os *bits* restantes de 3 ao 0 contém as *flags* específicas para cada tipo de mensagem, como pode ser visto na Tabela 2.

MQTT Control Packet	Fixed Header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT v5.0	DUP	QoS		RETAIN
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0
AUTH	Reserved	0	0	0	0

Tabela 2 –Tipos de *flags* MQTT 5

Fonte: IBM e Eurotech (2019)

É possível observar a existência de *flags* especiais na linha *PUBLISH*, são elas: DUP, *RETAIN* e QoS.

DUP (do inglês *Duplicate Delivery*), ocupa o *bit* 4 e quando ativa sinaliza quando o cliente ou o servidor tentam reenviar mensagens do tipo PUBLISH, PUBREL, SUBSCRIBE ou UNSUBSCRIBE.

RETAIN: quando ativa, a mensagem enviada ao servidor deve ser retida mesmo depois de entregue aos clientes do tópico.

QoS (do inglês *Quality of Service*), ocupa os *bits* 2 e 1 e indica o nível de garantia de entrega de uma mensagem *PUBLISH*. A Figura 2 mostra os diferentes valores que o QoS pode assumir e o seu significado.

QoS value	Bit 2	bit 1	Description
0	0	0	At most once delivery
1	0	1	At least once delivery
2	1	0	Exactly once delivery
-	1	1	Reserved – must not be used

Figura 2 –Valores da *flag* QoS

Fonte: IBM e Eurotech (2019)

O *byte* 2 da mensagem MQTT contém dados do cabeçalho variável (presente em alguns tipos de mensagem), contendo dois campos para nome e versão do protocolo respectivamente, e do *payload*, que pode armazenar diferentes tipos de informações dependendo do tipo de mensagem transmitida: *CONNECT*, *SUBSCRIBE* ou *SUBACK*.

CAPÍTULO 4: INSTALAÇÃO E CONFIGURAÇÃO DO MQTT NO LINUX

Um *broker* MQTT gerencia as mensagens transmitidas pelo sistema. Ele recebe mensagens de seus clientes e as envia para clientes destino, fazendo assim, uma comunicação automatizada entre as maquinas, sensores e aplicativos.

Existem diversas opções disponíveis para *broker* MQTT no Linux, dentre os mais populares, podem ser citados: *AWS IoT Core MQTT, Mosquitto, Mosca/Aedes e HiveMQ*.

Como vamos utilizar um *Raspberry Pi* em nosso trabalho, precisamos de um *broker* leve e eficiente. Por ser de código aberto, seguro, atualizado para o MQTT 5.0 e de fácil instalação, utilizaremos o *software* "*Mosquitto*" como *broker*.

4.1 INSTALAÇÃO DO *MOSQUITTO*

Antes de tudo, é importante definir um endereço IP estático para o *Raspberry Pi*, provendo uma conexão fácil e que não mudará mesmo depois de várias reinicializações. Também atualizar o sistema operacional utilizado, para isso, segue-se as seguintes linhas de comando:

sudo apt-get update e em seguida *sudo apt-get upgrade*

Com o sistema atualizado, instalaremos o *broker Mosquitto* com essa linha de comando:

sudo apt-get install mosquito

E em seguida o comando abaixo, para instalar o cliente do *software*:

sudo apt-get install mosquitto-clients

O *software* permitirá interagir e testar se o *broker* está funcionando corretamente no *Raspberry Pi.* Na instalação, o gerenciador de pacotes definirá inicialização automática do servidor junto do sistema. Para testar se a instalação foi efetiva, é possível digitar o seguinte comando:

sudo systemctl status mosquitto

A mensagem "*active (running)*" aparecerá no console caso o servidor esteja funcionando sem problemas.

REFERÊNCIAS

MARTINS, Ismael Rodrigues, ZEM, José Luís. “Estudo dos protocolos de comunicação MQTT e COaP para aplicações machine-to-machine e Internet das coisas”. Revista Tecnológica da Fatec Americana, Americana. v.3, n.1, p.64-87, mar./2015. Disponível em: https://fatecbr.websiteseguro.com/revista/index.php/RTecFatecAM/article/view/41. Acesso em 20 de outubro de 2021.

SANTOS, Bruno P.; SILVA, Lucas A. M.; CELES, Clayson S. F. S.; BORGES NETO, João B.; PERES, Bruna S.; VIEIRA, Marcos Augusto M.; VIEIRA, Luiz Filipe M.; GOUSSEVSKAIA, Olga N.; LOUREIRO, Antônio A. F. Internet das coisas: da teoria à prática. Departamento de Ciências da Computação, UFMG, Belo Horizonte, MG, Brasil, 2016. Disponível em: http://35.238.111.86:8080//xmlui/handle/123456789/329. Acesso em 20 de Outubro de 2021.

R A Atmoko et al 2017 J. Phys.: Conf. Ser. 853 012003

EMMET.; “INSTALLING THE MOSQUITTO MQTT SERVER TO THE RASPBERRY PI”. Disponível em: https://pimylifeup.com/raspberry-pi-mosquitto-mqtt-server/. Acesso em 01 de Março de 2022.