Arlen Nipper, Cirrus Link Solutions

MQTT's role as an IoT message transport

Message queuing telemetry transport's (MQTT) role as an Internet of Things (IoT) message transport began as an industrial communicator for a pipeline supervisory control and data acquisition (SCADA) system.

Message queuing telemetry transport (MQTT) has emerged as one of the dominant IoT message transports across multiple industries in the last five years. Considering that most cloud services provide native MQTT capabilities, more device manufacturers, software, and services are implementing MQTT-based products.

The genesis of MQTT

Adoption of MQTT by Facebook, cloud service providers, and many others in the information technology (IT) space might lead one to think that MQTT was invented targeting IT solutions, but the genesis of MQTT was driven by an industrial communication problem.

In 1997, Phillips 66 had installed one of the first transmission control protocol/internet protocol (TCP/IP)-based very-small-aperture-terminal (VSAT) systems in the market for use in its pipeline supervisory control and data acquisition (SCADA) system. Numerous challenges needed to be addressed to use this network infrastructure effectively. Poll/response protocols were the norm for any SCADA system implementation until this system was implemented.

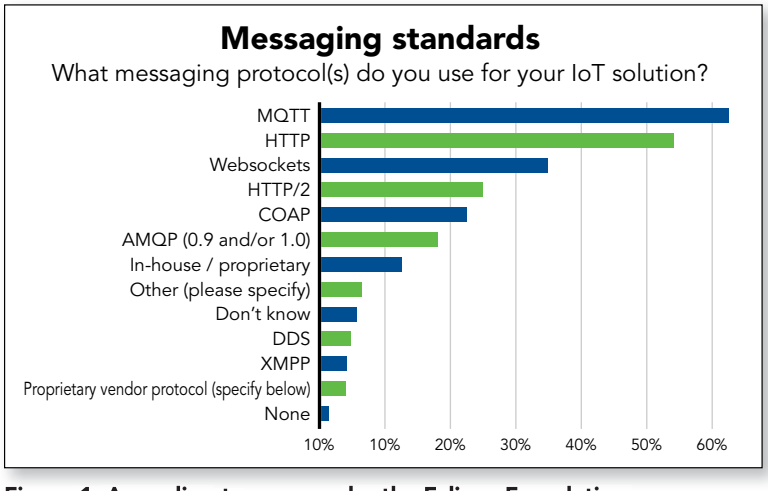

Figure 1: According to a survey by the Eclipse Foundation, message queuing telemetry transport (MQTT) is the most-used messaging protocol for an IoT solution. All graphics courtesy: Eclipse Foundation Inc.

However, due to the propagation delays inherent to VSAT communications, and the cost associated with continuously polling for process variables that may not have changed, Phillips was looking for a better way to optimize its network infrastructure.

By this time, information technology (IT) departments used message-oriented middleware (MOM) software to decouple applications from each other. They were efficient infrastructures that used message brokers to ensure applications that "published" information could be connected to applications that "subscribed" to that information. Information could be published on an exception basis to any application that had interest and was subscribed to that information.

The idea was to use this same type of infrastructure for a real-time SCADA system. The only problem was that MOM products on the market at that time weren't appropriate for use in the SCADA environment.

Based on these requirements, a project was started to develop a MOM specification that would be appropriate for use in these types of industrial environments. This ultimately led to the design of MQTT.

The original design goals of MQTT were that it would be simple, efficient, stateful, and open.

Simple. When MQTT first was being developed, the hardware platforms available on the market for remote edge computing were minimal; 8-bit microprocessors with 64 KB of memory were the norm. MQTT had to be simple to implement with minimal computing resources. Even in 2018, Arduino microcontrollers can provide complete MQTT communication stacks.

Efficient. Early VSAT system providers charged for every byte of information sent and received. The MQTT transport had to provide minimal overhead on the network. Once an MQTT session is established, there is only a 2-byte overhead in messages being published.

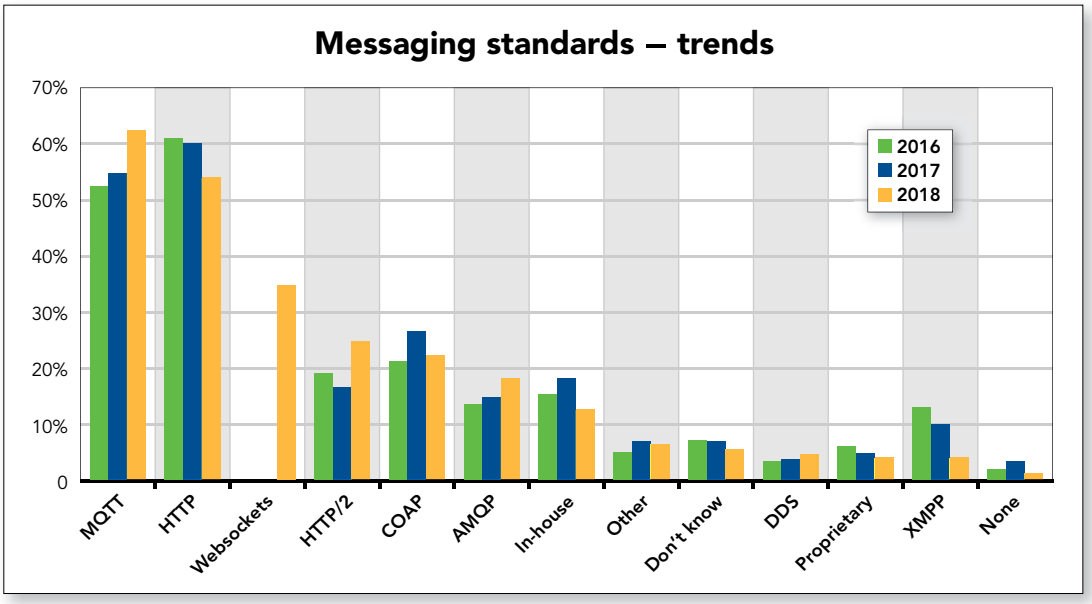

Figure 2: According to a survey by the Eclipse Foundation, MQTT is trending up since 2016 to over 62% use as a messaging standard.

Stateful. If a user is providing infrastructure for mission critical, real-time infrastructure then the "state" of the MQTT TCP/IP connection is critical. MQTT provides a mechanism called "continuous session awareness" that informs all clients that care about the real-time state information of the MQTT connections.

Open. In the late 1990's SCADA/DCS/Telemetry products were based mainly on proprietary legacy Poll/Response protocols. For MQTT to be useful to the industry as a whole it was understood that when it was released, it needed to be an open specification that anyone could implement for free.

Even with those criteria, it would be easy to assume a few important aspects are left out, including:

Security. A lot of people note the MQTT specification does not define any security. This is because the MQTT specification in based on top of TCP/IP. It always was envisaged that the latest TCP/IP security practices would be applicable to an MQTT infrastructure. This ranges from private networks where security isn't even required, to full transport layer security (TLS) certificates being used for connections. Since MQTT is a remote-originated connection, edge devices and clients don't even have to have any TCP/IP ports open, which is a huge reduction in the overall cybersecurity footprint.

Payload data format. MQTT is data agnostic when it comes to the information contained in an MQTT payload. It can be a binary message from a programmable logic controller (PLC), a JPEG image, an extensible markup language (XML) document or a JavaScript object notation (JSON) string. MQTT leaves the encoding and interpretation of the payload to the software provider.

Industrial-strength MQTT

As Internet of Things (IoT) solutions using MQTT started to migrate to more mission-critical Industrial IoT (IIoT) implementations, the market needed a specification that would allow MQTT-based vendors easy interoperability. Although the MQTT specification does not dictate any message topic namespace or data representation, one was needed for the IIoT space. The Sparkplug specification does that for the IIoT market.

The Sparkplug specification was developed to help define how best to get started using MQTT in a mission-critical, real-time application. The Sparkplug specification defines:

1. A well-known MQTT topic namespace so publishers and subscribers of information can know the topic namespace in advance for interoperability.

2. A binary payload optimized for industrial process variables. The Sparkplug specification acknowledges that industrial infrastructures don't have unlimited bandwidth and must work well over VSAT, radio, and cellular infrastructures.

3. How the "state" management in MQTT works and how to effectively use it in SCADA, distributed control system (DCS), and industrial control system (ICS) solutions to know the state of all MQTT clients in real time.

The Sparkplug specification and all of the reference implementation code written in C, Java, JavaScript, Python, and Node Red have been contributed to the Eclipse Foundation and to an open source project. **ce**

Arlen Nipper *is president/CTO, Cirrus Link Solutions. Edited by Chris Vavra, production editor,* Control Engineering, *CFE Media, cvavra@cfemedia.com.*

More ANSWERS

KEYWORDS: Internet of Things, message queuing telemetry transport (MQTT)

MQTT began as a communication protocol used by Phillips 66 to operate in tough industrial conditions.

MQTT's usefulness to engineers and manufacturers stems from ease of use.

MQTT has been enhanced to work in the Industrial Internet of Things (IIoT) era.

CONSIDER THIS

For what applications do could you use MQTT?

ONLINE

Read this story online at www.controleng.com for more information about the author and related stories about MQTT.