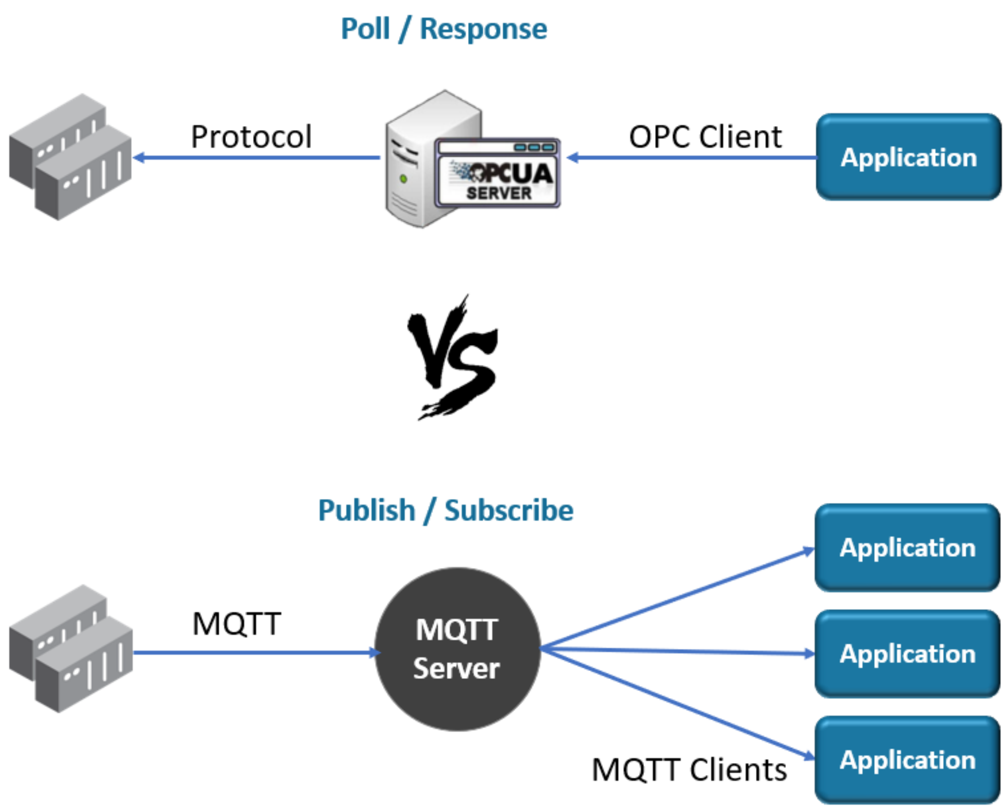
IIoT Protocols: Comparing OPC UA to MQTT



Introduction

The Internet expanded rapidly thanks to two technologies. First HTTP, a data exchange protocol, and then HTML, which was used to define the data sent by HTTP. Both technologies were needed, and the explosion of the Internet was made possible, and interoperable, by the uniform adoption of these common standards.

Industrial IoT has not yet enjoyed this type of uniform adoption that leads to rapid growth at scale. There are several different architectures and protocols that can be used to gather data on the factory floor and deliver it to data consumers across the enterprise. Some common options are CoAP, DDS, HTTP, MQTT and OPC UA. For the purpose of this discussion we'll compare two options on opposite ends of the spectrum – OPC UA and MQTT/Sparkplug.

OPC UA is the next generation standard from the OPC Foundation, released in 2008 as an update to the original OPC interoperability standard for secure and reliable exchange of data in industrial automation. OPC classic was released in 1996 was designed to abstract PLC-specific protocols into a standardized interface to allow SCADA systems to access the data. OPC is built on a client/server architecture. The OPC server converts the hardware communication protocol, then any program that needs to connect to the hardware becomes the OPC client software.

MQTT is a transport protocol invented in 1999 as a lightweight, publish-subscribe network protocol that allows for multiple data consumers and is designed for constrained devices and low-bandwidth, high-latency or unreliable networks. MQTT is based on a message-oriented middleware approach. Sparkplug is an open source software specification that provides MQTT clients with a framework to integrate data – defining a Topic Namespace, State Management, and Payload to make the data interoperable for IIoT applications.

Today, many manufacturers or users of process control monitoring have made a choice based on the existing architecture in their environment. If they have a SCADA system, they tend to use OPC or OPC UA. However, new manufacturers or those looking to digitally transform should consider MQTT/Sparkplug to solve modern challenges and adopt an IIoT solution that can handle any number of data producers and consumers across the enterprise with ease.

A Deeper Look at OPC UA

Benson Hougland, VP Marketing & Product Strategy for Opto 22, an industrial control products and Internet of Things platforms manufacturer, remembers some of the history of OPC from the hardware vendor perspective. "Back in the mid-90s we were developing PLCs and we had a challenge in industrial automation – we needed to share operational data with other software systems, but every vendor had their own proprietary communication protocols."

As a result, four companies including Opto 22 worked with Microsoft to try to improve the situation in industrial automation. They became the original authors of OPC.

"OPC has its place on the shop floor and has been around for many years," said Hougland. "It was developed for Windows, traditionally requires a dedicated Windows PC, and does a good job of exchanging OT data with legacy systems on a local area network."

OPC uses a client/server architecture, ideal for connecting PLCs to SCADA and MES systems where OPC and OPC UA are already available. However, there was a call for scalability and a platform-independent architecture, which led to the development of OPC UA.

According to the OPC Foundation, "With the introduction of service-oriented architectures in manufacturing systems came new challenges in security and data modeling. The OPC Foundation developed the OPC UA specifications to address these needs and at the same time provided a feature-rich technology open-platform architecture that was future-proof, scalable and extensible."

OPC UA aims to expand interoperability to the device and enterprise applications and recently adds publish/subscribe to the original client/server communications infrastructure. The spec also attempts to remove the requirement for a dedicated Windows PC by permitting a server to be embedded into an edge product. The OPC Foundation open sourced the OPC UA standard to increase adoption.

OPC has several limitations. OPC UA tries to solve some of these problems but is extremely complicated to implement. Consider the following issues before implementing an OPC or OPC UA architecture:

Complicated: The most common complaint about OPC UA, or OPC for that matter, is how complicated it is to implement. The original spec is over 1000 pages and OPC UA is 1250 pages. Many companies do not implement the full OPC Server, and even if they do it also requires routers, firewalls, and VPN. New devices must be configured and connected rather than auto-discovered. "Because OPC UA is so complex it does not often get completely or accurately implemented," says Hougland.

Heavy: When an OPC server talks to a client, since the subscriptions are not reported by exception but traditionally use a poll/response protocol (other than a very few modern OPC UA implementations), they are very heavyweight. A lot of devices that support OPC cannot handle a lot of subscriptions because they do not have enough storage. This limits the number of data consumers on the system. In addition, even though it is heavy, OPC data only comes with basic tags and values – no meta data or objects.

Inflexible: The shop floor is becoming increasingly varied – with both modern and legacy devices, different operating systems, network architectures, and more. OPC has a hard time handling these varied data structures and heterogeneous devices. Classic OPC and OPC UA implementations usually include an OPC server that talks native protocols to the devices, and then clients come and get the data. The focus is on OT data and the solution falls short for big data analytics applications, since it only collects OT data and not IT.

Expensive: When OPC UA is fully implemented it is very expensive and heavy. The architecture requires embedding an OPC UA server into products which increases the cost and time-to-market, increases the footprint size, CPU utilization, development costs and ongoing support costs.

Lack of Vendor Support: OPC has been around for many years, but many IIoT technologies do not come with native support for OPC connectivity. Microsoft is the only cloud vendor that uses both OPC UA client-server connections as well as the new OPC UA publish-subscribe connections. Hardware vendor participation has been lackluster and there are only a few OPC

UA software clients available. Typically a shop floor does not have a lot of assets that natively support OPC UA, so they use an OPC UA server to serve the data up to OPC UA clients.

Struggles with Multiple Data Consumers: OPC architectures struggle to supply all of the data to multiple data consumers. An OPC UA server does provide one-to-some capabilities but does not do the real decoupling needed for one-to-many. In addition, most implementations don't include meta tag data, which once again means it falls short on getting data to IT functions in a usable format.

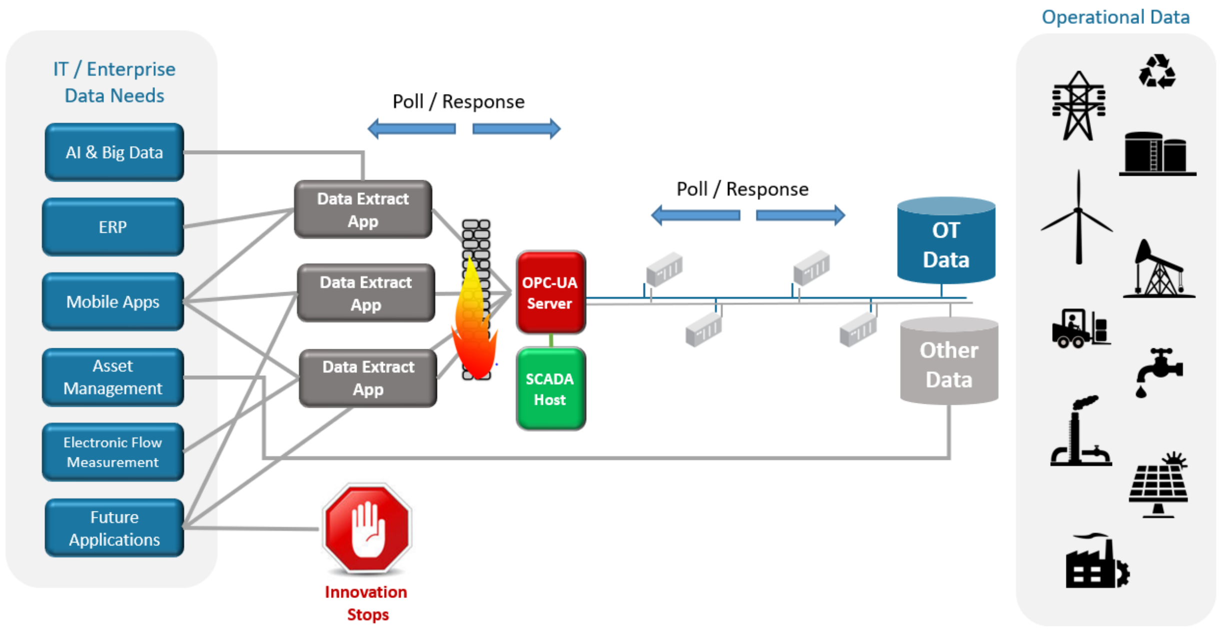


Figure 1: Traditional SCADA systems isolate OT data and can't handle multiple data consumers

A traditional OPC system is shown in Figure 1, where SCADA owns the data path that was built for operations, or OT data. When new consumers request OT data, new application or custom code is written to get the data out of SCADA. As new data consumers are added, a brittle enterprise of applications is built that is costly to manage and becomes too complex to change. The organization is trapped from moving to new technology without tremendous costs and operational disruption.

Use Cases: Traditional discrete and process manufacturers are the ideal use case for OPC. Without any distributed assets it is harder to make a case for switching OPC out for MQTT. If the shop floor is already built as a SCADA system and it is working, then OPC or OPC UA work as well. However, any shop floor that wants to implement advanced technologies such as machine learning, AI, or predictive maintenance will have a hard time with OPC UA. It was not made for these modern challenges. OPC UA is a good solution for tying PLC and sensor data into existing SCADA and MES systems, but today's shop floor needs a more flexible protocol to successfully transfer IIoT data northbound across the enterprise.

A Deeper Look at MQTT

Inductive Automation, an IIoT and SCADA software platform developer, was involved in writing the first OPC UA server that could run on Linux or Windows. “OPC has always been complicated,” says Travis Cox, Co-Director of Sales Engineering for Inductive Automation. “MQTT was implemented to be simple and lightweight with low overhead and high performance. Interoperability is easier with MQTT than OPC.”

MQTT was invented to serve multiple data consumers and multiple data producers. In order for Digital Transformation to be successful, data must be decoupled and provided over an enterprise-wide solution architecture. MQTT allows for multiple data consumers (Figure 2). Companies can publish the data from a manufacturing asset and multiple applications can consume it, all at the same time.

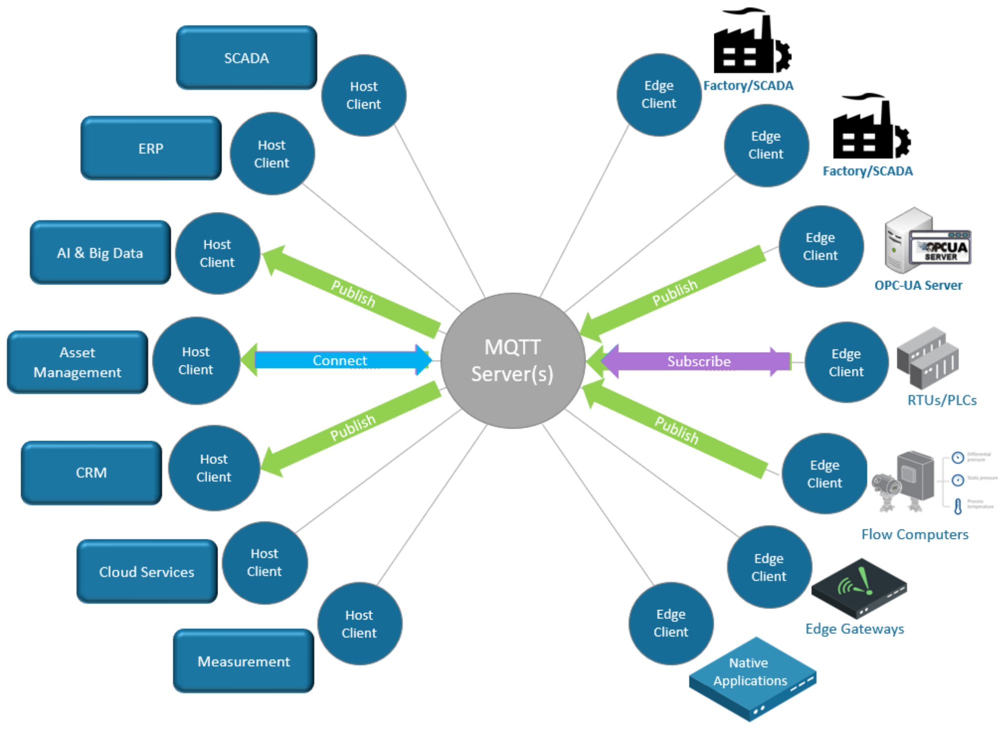
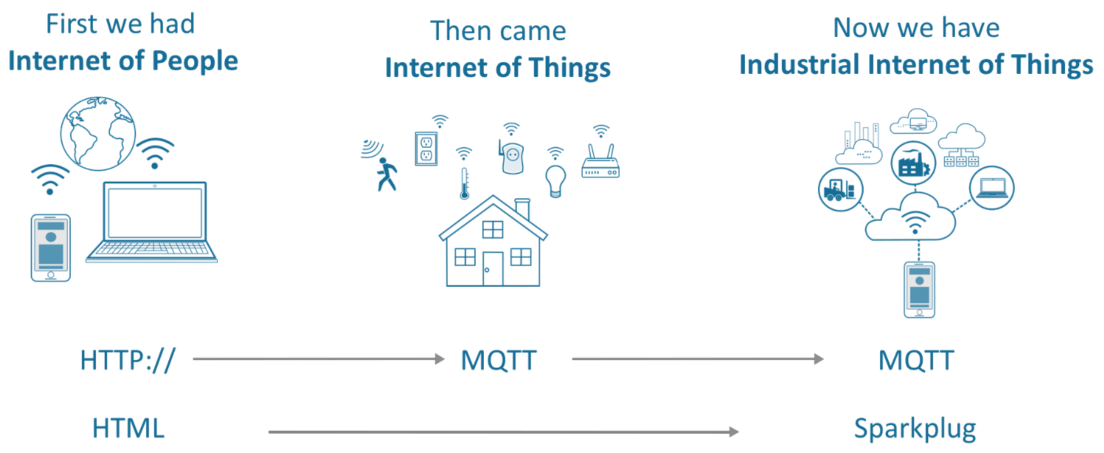


Figure 2: The basic MQTT architecture allows for unlimited clients over a publish/subscribe protocol.

Cirrus Link recently created a specification within the Eclipse Tahu project called Sparkplug that defines how to use MQTT in a mission-critical, real-time environment. Sparkplug does for MQTT what HTML did for HTTP – defines the data.



Sparkplug defines a standard MQTT topic namespace, payload, and session state management for industrial applications while meeting the requirements of real-time SCADA implementations. Sparkplug is an open standard that is license-free to use and builds on 20 years of experience of how to use MQTT. The Sparkplug B specification provides the context data needs to define a tag value for use with OT, also providing data to IT, making it 100% self-discoverable and easy to consume.

Utilizing MQTT with the open-standard Sparkplug data representation provides the tools for organizations to build a cost-effective solution for Digital Transformation across their enterprise. With minimal risk and cost, MQTT allows OT data to be consumed with simple configurations on proven software tools that securely bridge the OT/IT gap and provide contextual information for the data scientists to use Big Data Analytics, ML, and AI to gain insight and increase productivity and profit.

The following points explain how MQTT solves the pain points not addressed by OPC UA.

Simple: The MQTT spec is 80 pages and Sparkplug adds another 60. Developers can follow the spec and implement it on their own, within a few days or weeks. MQTT is very easy to implement since devices can be added and auto discovered. "If I were to ask a few college students to get MQTT running, they could do it easily," says Cox. "There is tons of reference code and it is simple, flexible, and powerful."

Lightweight: MQTT reports by exception, minimizing the data footprint and leading to more efficient communications. The protocol overhead is extremely small, the smallest packet has only 2 bytes of overhead. With Sparkplug, MQTT also includes the essential meta data with no additional work required, and still keeps it lightweight.

Flexible: MQTT is based on a pub/sub model that decouples data publishers from consumers, which means subscribers do not need to know who provides the information to which they are subscribed. The message can be in any data format for the payload, such as JSON, XML, encrypted binary or Base64, which gives a lot of flexibility to the protocol.

Cost-Effective: IIoT powered by MQTT provides a cost-effective solution for access to data on brownfield devices. MQTT can transport the data from a sensor, to a device (such as a PLC), to an Edge gateway, and then up to the SCADA/MES system on the factory floor.

Secure: MQTT provides security in several ways starting with client usernames and passwords required to login. MQTT uses the most current TCP/IP layer security available which today is TLS. Next the connections are remote originated, so no ports are open in the field and only one port at the main firewall is connected to the broker. Access Control Lists (ACLs) only allow known remote devices to connect.

Vendor Support: The number of vendors natively implementing MQTT-Sparkplug, both on the hardware and software side, is growing rapidly. All of the leading cloud vendors, IoT platforms, edge computing platforms, big data and other third-party applications support MQTT. Progressive cloud providers are implementing Sparkplug to give auto discovery data modeling.

Supports Unlimited Data Consumers: Moving to a publish/subscribe model with MQTT enables the transition from a one-to-one to a one-to-many approach, encouraging innovations while making it easy to adopt new technologies. Data producers publish the data in Sparkplug B format to an MQTT server. The MQTT server enables those who have secure access to subscribe to the data. The OT application will subscribe to the data instead of polling for it.

Use Cases: Customers in oil & gas, telemetry and energy were the first to see the benefits of MQTT where it was originally invented. Polling widely dispersed assets for their data does not make sense in today's world of IIoT. MQTT allows any data consumer to easily subscribe to the data and pub/sub saves bandwidth and simplifies the solution. However, discrete and process manufacturers who may be committed to OPC or OPC UA can also realize the benefits of MQTT when it comes to sending data to the cloud and integrating with big data applications.

OPC UA and MQTT Can Work Together

It is important to note that OPC UA and MQTT can actually work together harmoniously. They may be polar opposites in the way they move data around, but there are still old devices that need an OPC server to share data and there is a way to use MQTT to overcome the challenges presented. With a sensor connected to a legacy PLC, IoT platforms like Ignition from Inductive Automation can connect and translate that data into MQTT, move it across any type of network in a pub/sub model, and then either send it to the cloud and enterprise apps, or some IoT platforms will translate it back into OPC for legacy OPC clients.

"We have to support both OPC UA and MQTT/Sparkplug," says Cox. "We bridge between the old and the new. Some people think OPC UA has its place on the shop floor and MQTT has its place in dispersed asset use cases, but really MQTT is the future for any manufacturer of any kind to send data across the enterprise for modern IIoT solutions and third-party applications."

"Twenty-five years after we co-authored OPC, with the arrival of IIoT, Digital Transformation and Smart Manufacturing, the world has changed," said Hougland. "OPC was designed to solve a problem on the plant floor with a constrained network and little security. It works fine on a private network, but that's not the world of today's connected plant floor – and definitely not IIoT."

OPC still has a place on a constrained, private network and for a point-to-point connection where multiple data consumers are not necessary. OPC still has a place for a customer who is not interested in adding new technologies and new capabilities.

"We wouldn't be where we are without OPC," says Cox. "It has a stake hold on the OT side and it's going to be needed for quite some time. There is a lot of legacy equipment out there, and OPC servers make it easier for platforms like Ignition to convert that to MQTT."

However, if a manufacturer has a choice of purchasing devices that support pub/sub with Ignition Edge or native support for MQTT/Sparkplug, an MQTT implementation requires less effort, less money, and less time than OPC. Using MQTT allows customers to embrace newer IoT technologies and software like big data, machine learning, and more. Just as HTTP and HTML worked together to enable the rapid expansion of the Internet, so does MQTT and Sparkplug, providing users with a modern option for interoperability of data on the shop floor.