Whitepaper

Data exchange in the age of the “Industrial Internet of Things”

Markus Weishaar | Product Manager IIoT
Dunkermotoren GmbH

As a result of the Industrial Internet of Things (IIoT) being established as a key technology of the fourth industrial revolution, data generated by machines and components are increasingly coming to the fore for companies. Not without reason are data being described as the new commodity, or gold, of the digital age. This is because data analysis enables the acquisition of information about associated products and processes which, in turn, forms the basis for productivity increases or even brand-new business models. For example, by analyzing the operating data for motors, service-related business models, such as predictive maintenance, can be derived on the one hand, which on the other hand, minimizes production down times, thus increasing productivity. But how do these data find their way from the device generating them to the place where they are analyzed? This whitepaper addresses this question and examines the most conventional communication standards in the context of the IIoT.

Nowadays, there is no doubt that classic fieldbus systems, such as DeviceNet or Profibus, will be replaced by industrial Ethernet buses, such as Profinet, EtherCAT or Ethernet/IP, in the medium term. This is also borne out by the annual review of the market share of bus systems conducted by HMS, which shows the first indications of a decline in classic fieldbuses in 2019 (Figure 1). It is thus clear at any rate that in automation, just as in IT, Ethernet is also set to become the transmission medium in future as regards wired connections. Nevertheless, with industrial Ethernets, proprietary standards continue to dominate field and control levels. At present, this further complicates cross-manufacturer communication. However, even here, there appears to be light at the end of the tunnel as a result of the comprehensive support of OPC UA as a uniform standard, in the move towards shared and open horizontal and vertical communication. This is crucial, as standardized and open communication between devices and systems is the greatest challenge, as well as the greatest enabler, for the Industrial Internet of Things. This is because only a non-proprietary and no-tier form of communication provides machines and plant engineers with the opportunity to select and combine the best technologies for their specific application. This is the only economically viable way for data to be transmitted from all field devices to higher level analysis software, as they do not need to be gathered and interpreted from a wide range of proprietary systems.

In addition to these communication standards, which originate in automation, the policies originating from IT for data exchange via API (application program interface) or via protocols such as MQTT and AMQP are becoming increasingly important in automation, as the issue is increasingly one of transmitting data from automation devices to IT systems.

The individual technologies, their advantages and disadvantages, are detailed and illustrated below:

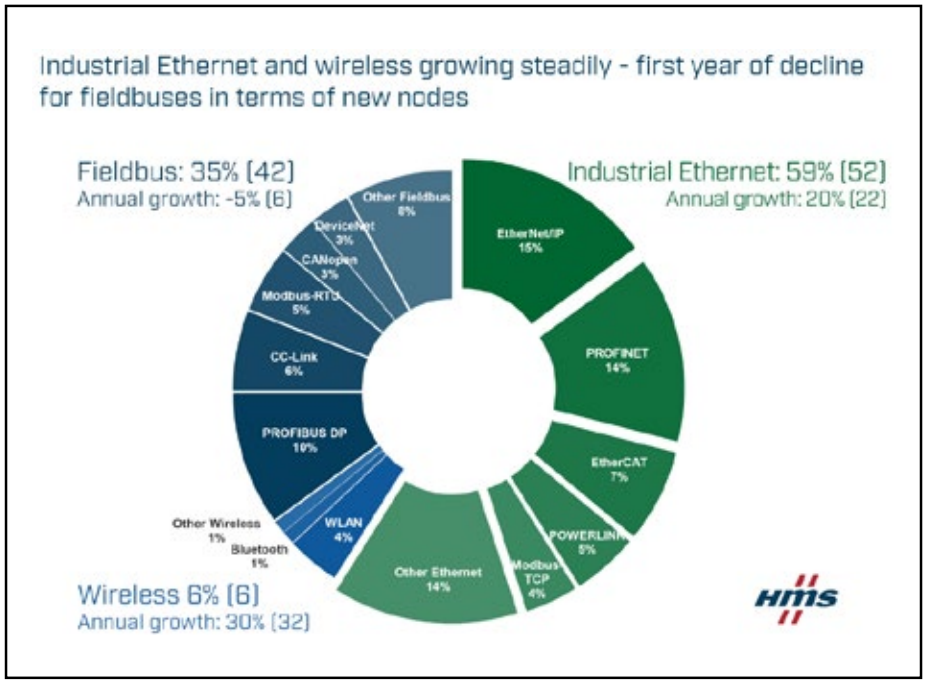

Figure 1: Market share industrial networks[1]

Author: Markus Weishaar
Date: 02.18.2020

Industrial Ethernet:

Industrial Ethernets are manufacturer-driven proprietary communication protocols, based on the Ethernet standard IEEE 802.3, extending this to include real-time capability and various additional control functionalities, such as “motion”, for example. The three most widespread protocols in use are EtherNet/IP (Rockwell), Profinet (Siemens) and Ether-CAT (Beckhoff). The extent to which the individual industrial Ethernets rely on the conventional Ethernet varies from one system to another. The first two layers of the OSI model, standardized according to IEEE 802.3, these being the physical transfer layer and the MAC address, are used by practically all protocols. However, based on this, the extent to which the conventional Ethernet‘s standard IP, TCP and UDP protocols are accessed depends heavily on the respective industrial Ethernet. EtherNet/IP is the closest to the conventional Ethernet, as, in this case, the standard IP, TCP and UDP protocols are also a fixed component of the protocol stack and only above the transport layer does the EtherNet/IP rely on specific CIP protocol. Here, Profinet is a separate protocol stack above the MAC address, but, when establishing a connection and with acyclical services, still offers the option of falling back on UDP/IP. By contrast, EtherCAT sits above layer 2 and is a completely independent protocol, which only uses the Ethernet frame. This is one reason for the outstanding performance of EtherCAT regarding the latencies for which it was developed, but also because EtherCAT is the furthest away from the conventional Ethernet. Protocols are, in principle, open and can also be implemented by third-party suppliers in their devices. The protocols deploy their full performance, but usually only in the ecosystem of the manufacturer that developed the protocols. It thus becomes apparent that licenses are often required and particular hurdles exist specifically when integrating master capability. The “openness” thus serves more as a means of integrating the devices of third-party suppliers into an individual ecosystem and to tie users in to the individual ecosystem. Based on the development cycle of around 15 years in automation, industrial Ethernets will continue to play a key role in the coming years, even if an overarching system, such as, for example OPC UA, were to be established. The assumption is thus supported that, according to the HMS study, industrial Ethernets have only overtaken classic fieldbuses, as regards the number of new nodes, since 2018, some 15 years after their initial implementation. Furthermore, classic fieldbuses only began to decline in 2019.

OPC UA:

OPC UA stands for Open Platform Communication Unified Architecture. Therefore, OPC UA should not be seen simply as an additional communication protocol, but rather as a communication framework that, besides data transmission, also details the significance of and access to data, as well as including security mechanisms. Security here is implemented by means of certificates and certificate and access rights management, which are included in OPC UA. Nevertheless, it must be noted that OPC UA is not secure per se, but rather it includes the necessary mechanisms and properties which can be used as the basis for implementing secure communication. OPC UA is based on the premise that devices are mapped in the form of information models. This means that devices are described in objects with their associated variables, methods, events and their connection to other objects. For example, in its simplest form, a motor can be an individual object with variables, events and methods. However, it can also comprise various objects, such as, for example, current controllers, position controllers and speed regulators. Consequently, any complex components and machines can be mapped semantically and it is possible for an OPC UA application to be able to understand these semantic models, without knowing them in advance. To ensure this functions across manufacturers, OPC UA is based on a service-oriented architecture (SOA), which defines access to information models via standardized services. In the basic implementation, OPC UA includes the information models of Data Access (DA), Alarms & Conditions (AC), Historical Access (HA) and Programs, as well as Services Browse, Read/Write, Method Access and Subscribe to

Author: Markus Weishaar
Date: 02.18.2020

individual variables. Furthermore, OPC UA functionality can be extended via industry-specific information models, known as “companion specifications”. To also take account of the various distinguishing features of individual manufacturers in industries such as e.g. drive technology, there is also an option to map non-standardized functions via the manufacturer-specific extensions in OPC UA (Figure 2).

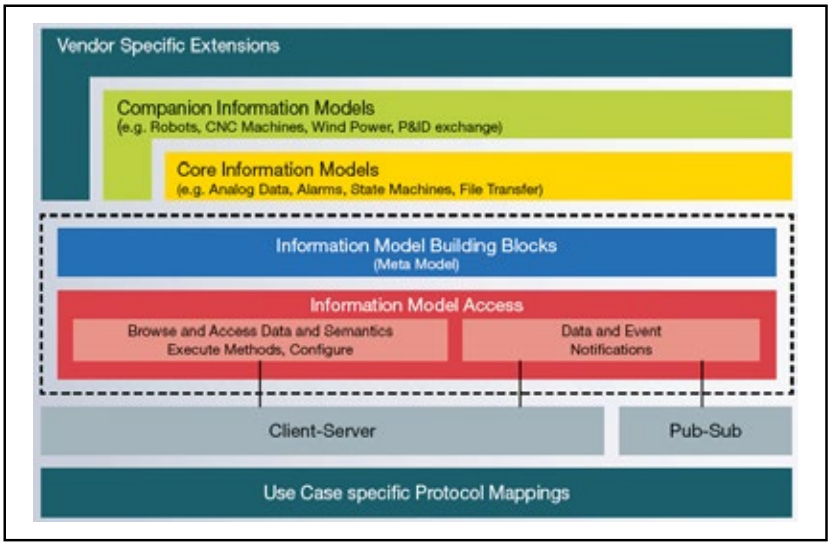

Figure 2: OPC UA Structure[2]

Fur actual communication, OPC UA provides two types. Server/Client and Publish/Subscribe. Both communication types can be used in parallel with an OPC UA application and any application can also assume any role (Figure 3). OPC UA thus covers the usual direct connections in automation via Server/Client and the usual indirect connections in Cloud connectivity via Publish/Subscribe. With Publish/Subscribe, OPC UA relies on the embedding of common standards, such as MQTT or AMQP. In the Publish/Subscribe context, a variant with UDP as the transmission protocol is also supported at field level. At the transport layer of communication, OPC UA continues to be based on Internet protocol (IP) and thus always requires an Ethernet-based network infrastructure. In its current form, OPC UA is therefore an ideal addition for industrial Ethernets to transfer data in parallel with control communication to higher-level systems for analysis and monitoring. Work is currently in progress on enhancing OPC UA with Time-Sensitive Networking (TSN), and thus to enable deterministic real-time communication. This approach, as well as 5G, also forms the basis of the Field-Level-Communication (FLC) initiative of the OPC foundation, which is working on bringing OPC UA communication to field level. OPC UA as an Ethernet-based and deterministic communication standard at field level thus has the potential to enable standardized, cross-manufacturer communication. Even for local system architectures with no distinct control level, this is a promising technological step, as field devices can communicate in the same way with each other and with control and management systems. As detailed, the features of OPC UA make it the ideal toolkit for cross-manufacturer and cross-level communication. Nevertheless, there are some limitations to bear in mind. The high degree of abstraction required by OPC UA to enable the

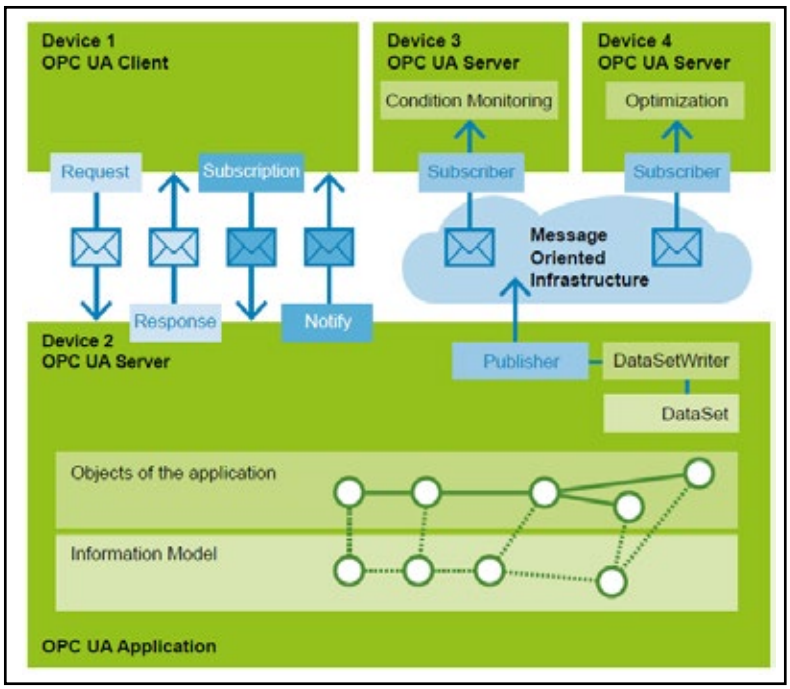

Figure 3: OPC UA Communikation[3]

Author: Markus Weishaar
Date: 02.18.2020

generic approach, naturally makes entry difficult, as well as the analysis of the connection itself. Furthermore, all the features included in OPC UA also result in a corresponding demand on the hardware used to run an OPC UA application. For this reason, reduced OPC UA profiles are also available, which do not support any security mechanisms or which only permit a connection at the lowest level, without method accesses and subscriptions.

MQTT:
Message Queuing Telemetry Transport is a lightweight protocol for data transmission. At best, data packets of just 2 bytes can be implemented, which is particularly advantageous for a large number of devices and messages. This benefit, together with the ease of implementation of MQTT, has thus resulted in very heavy use of MQTT in the IoT environment in recent years. However, it must be noted that, in contrast to OPC UA, MQTT is purely a transmission protocol and does not provide any enhanced framework with functions such as semantic data description or security mechanisms. This means that security measures to safeguard the connection must be implemented separately and the type of data concerned and how the data should be understood must be declared on both sides of the communication. The structure of MQTT is an open Publish/Subscribe protocol for indirect 1 to n communication. This means that a publisher based on events sends messages on a specific topic to what is known as a broker. The broker forwards the corresponding message to all subscribers who have subscribed to the associated topic. With regard to a motor, this may mean, for example, that under the topic “Diagnosis/Overcurrent”, the motor’s serial number and associated value is communicated to a broker when the preset limit value is exceeded. The broker then forwards the message to all subscribers, such as mobile terminals belonging to service engineers, control systems, or cloud applications. When configuring the Publish/Subscribe communication, MQTT offers some more useful features. Retained Messages, for example, enable the last message sent on a topic to remain with a broker, so that it can be communicated to a new subscriber as soon as they subscribe. There is also a Last Will message, that a publisher can hold with the broker. This will be sent to all subscribers if a device is no longer connected and if any previous logout or unsubscribe action has failed. Even if the subscriber loses their connection, the Persistent Session feature provides the opportunity for the missed messages to be buffered (or “cached”) in the broker. They can then be communicated to the subscriber next time they log in. As a last resort, various Quality of Service settings offer three more options to ensure that sent messages reach one, more than one or at least one subscriber. Consequently, MQTT is a protocol that is easily controlled and ideally suited for implementation on devices with limited resources. It can also guarantee that data reach their destination in the event of any loss of connection. However, parties on both sides must be aware of the data communicated and there must be other means of ensuring data security, particularly for the broker.

AMQP:
Alongside MQTT, Advanced Message Queuing Protocol is the most widely used communication protocol in the IoT environment. Like MQTT, AMQP works with a broker and the Publish/Subscriber principle. With AQMP, each subscriber has a queue in which messages from the broker with subscribed topics are held. The messages remain in a queue until the subscriber confirms that they have received the message. The queues are thus also a buffer for messages in case a subscriber is not always connected. If a message cannot be communicated to a recipient, the Publisher receives a corresponding message. However, in addition to Publish/Subscribe, AMQP offers the following types of transmission:

»Fanout, where the broker transmits a message to all connected queues.

Author: Markus Weishaar
Date: 02.18.2020

» Direct, where a fixed connection between a subscriber and a queue can be established by means of an identifier.

» Headers, where messages in the broker are distributed via message headers instead of identifiers, which offers more rule-creation options compared with Direct transmission.

With AMQP, messages can also be supplemented with metadata, which describe message data in the form of attributes and which can be used by the recipient.
The main way in which AMQP differs from MQTT is thus the extended range of functions offered by AMQP for message transmission. However, implementation of AMQP also requires more effort and more resource. The smallest possible packet size with AMQP is 60 bytes. It is therefore worth considering whether the extended functionality offered by AMQP is actually needed, or whether the simpler MQTT solution would suffice.

API/REST API:
Application Program Interfaces originate from the concept of subdividing programs into function-based modules. The individual modules provide other modules with their public data via APIs and use APIs to retrieve the necessary data from other modules. APIs are structures with a number of variables, which are described by the associated module and which can be read by other modules. Consequently, they decouple the module's "private" code from the outside world. This makes it possible to create modules that are easy to maintain and to identify incorrect code more quickly, as each module can be tested in its own right by means of the API description with the intended commands and by checking the anticipated results. APIs can thus generally be precisely customized to suit each individual application. However, they are manufacturer-, application-and module-specific, and are not standardized. APIs therefore need to be described in terms of how they are reached, which variables they include and whether the variables have write access or are read-only. Even when exchanging data with other manufacturers via APIs or with public APIs, the detailed description is important, so that "external" programmers know how they can use the interface, as they have no knowledge of how the API's underlying application works.

REST
stands for Representational State Transfer. REST is not a particular standard or protocol. It is an architectural approach for communication in distributed systems. As REST is not a specified standard, there are consequently no details on what compliant implementations should look like. However, there are six architectural principles (constraints) to be respected. In technological terms, REST also relies on what is already there. Therefore, with data transmission, HTTP/S is often used as the protocol and XML (Extensible Markup Language) or JSON (Java Script Object Notation) are often used as the data format for information. As REST was developed in 2000, at a time when the Internet had its momentous breakthrough, it was actually the Internet that provided REST with a large part of the infrastructure required and most Web services are based on REST. The six architectural principles defined by REST are as follows:

REST is built on a client-server model with strict separation of data storage and user interface. This means that user interfaces as clients can easily be adapted to different individual framework conditions, whilst data storage as a server is easily scalable by means of a standardized structure.

Messages must be stateless. Consequently, a client query to the server needs to be self-contained and include all information about the application status. The context of the message must therefore always be provided, as there are no existing sessions with REST and the server cannot otherwise

interpret these. This principle also ensures easy scalability, as various messages from the client can be processed by different servers.

The client has the option to buffer (or "cache") the server's response for another identical query, if this is flagged accordingly. This helps to reduce network traffic and increase network efficiency. However, there is a risk of the client accessing obsolete data.

REST relies on a standard interface between all clients and servers with standard protocols, data formats and methods for access. The use of standard interfaces is usually accompanied by performance losses, as all data must be converted into a standard format. However, these losses are usually accepted willingly for the sake of simple architecture and usability. One example structure would be the use of HTTP/S the transmission and JSON as data format, as well as the following common methods:

» GET - requests data from the server
» POST - transmits data to the server
» PUT/ PATCH - amends existing data on the server
» DELETE - deletes existing data on the server

REST specifies an architecture in a layer system with a clear hierarchical structure and delimitation between the layers. This approach enables greater abstraction, thus giving the user access to various underlying architectures via a standard interface layer, without the user being aware of this. It is therefore possible to encapsulate legacy systems as a layer, for example, and make them accessible via "new" interfaces. This results in increased security and usability. However, abstraction also means increased overheads and latency times as a result of communication via a number of layers. As the sole optional principle with REST, "code on demand" offers the opportunity to download or transmit executable code to the client via the API. This gives the option of modifying or enhancing a client's function independently of their own code.

Conclusion:
To summarize, we can say that even in the future there will not be one communication standard across all levels, although the variety will reduce significantly. It is most likely that OPC UA would have the potential to enable vertical and horizontal communication at and across all levels. However, for such a scenario, OPC UA also has certain requirements for hardware in the form of storage, processing power or even cryptological chips, most of which current embedded devices may not yet be able to satisfy. Also, as regards TSN, it is not yet possible to anticipate precisely when and in what form OPC UA can be implemented at field level as real-time bus. At present, and over the coming years, the heterogeneous industrial Ethernet fieldbuses will continue to dominate at field level, as here too, the investment cycle of approximately 15 years for machines and systems must be taken into account, during which time existing systems will gradually be replaced by new ones. Here, it is more likely that OPC UA will come into effect as a standard Ethernet interface for EDGE so that there is no need for each protocol to connect for each device. But also in the field of Cloud communication, Publish/Subscribe protocols, such as MQTT or AMQP data exchange via REST APIs, have meanwhile become established as a quasi standard for a number of applications, which makes replacement in the short-term unlikely.
Based on this situation and the given constraints, the communication architecture detailed below appears likely for the future:
Data from field devices are transmitted independently of the industrial Ethernet fieldbus via OPC UA or alternatively also directly via the respective industrial Ethernet" fieldbus to the EDGE. There,

Author: Markus Weishaar
Date: 02.18.2020

the incoming data are implemented via software adapters in the necessary Internet protocols, such as MQTT or AMQP, or directly transferred by means of a REST API. Which way will be used exactly depends heavily on other factors such as where data are transmitted to or which IIoT ecosystem is used to process the data. At Cloud level, data exchange via REST API currently looks most likely. When it comes to transmission, various technologies will also be used in future on various layers, albeit with greater standardization within the levels. In contrast, as regards data format, greater standardization will emerge across all levels. Here, the semantic data description in JSON specified by OPC UA currently has the potential to consistently become the quasi-standard.

References:

1 https://www.hms-networks.com/de/news/pressemitteilungen-von-hms/2019/05/07/marktanteile-industrieller-netzwerke-2019-aus-sicht-von-hms
2 OPC Unified Architecture, Interoperability for Industrie 4.0 and the Internet of Things; Version 10 INA; OPC Foundation, 2019; pg. 24
3 Industrie 4.0 – Kommunikation mit OPC UA; VDMA; Fraunhofer IOSB-INA, 2017, pg. 13

Your Contact For Public Relations:
Janina Dietsche | janina.dietsche@ametek.com
Phone.: +49 (0)7703/930-546

Author: Markus Weishaar
Date: 02.18.2020