Choosing a Connectivity Standard

When it comes to selecting from the standards that cover the IIoT, the choice should depend on understanding the technology and its application, according to **Dr. Stan Schneider**

The IIoT covers many industries with very different use cases. The connectivity technologies and standards that target these applications are themselves diverse. In fact, the IIoT space is so big that the technology options barely overlap.

The architecture challenge in the IIoT space is not one of choosing among overlapping standards that may each be able to reasonably solve a problem but rather understanding the technologies, comparing the intended use to the application, and choosing the one that best addresses the particular challenge.

Stretching a technology out of proportion can make anything work. But, that will result in a lot of extra work and an awkward design.

Before the Industrial Internet Connectivity Framework (IICF), people assumed that competing standards met overlapping requirements in the IIoT connectivity space. It turns out, however, that those connectivity standards do not overlap. Most applications will not be a perfect fit and so must adapt.

After all, while connectivity technologies all move data they are nonetheless very different.

Since the connectivity options are so different, in most use cases, there tends to be no choice in the connectivity technology, but this lack of overlap in the IIoT space actually helps to makes an architect's task much simpler.

It's possible, however, to ask a few very simple questions for each technology option and quickly narrow the choices.

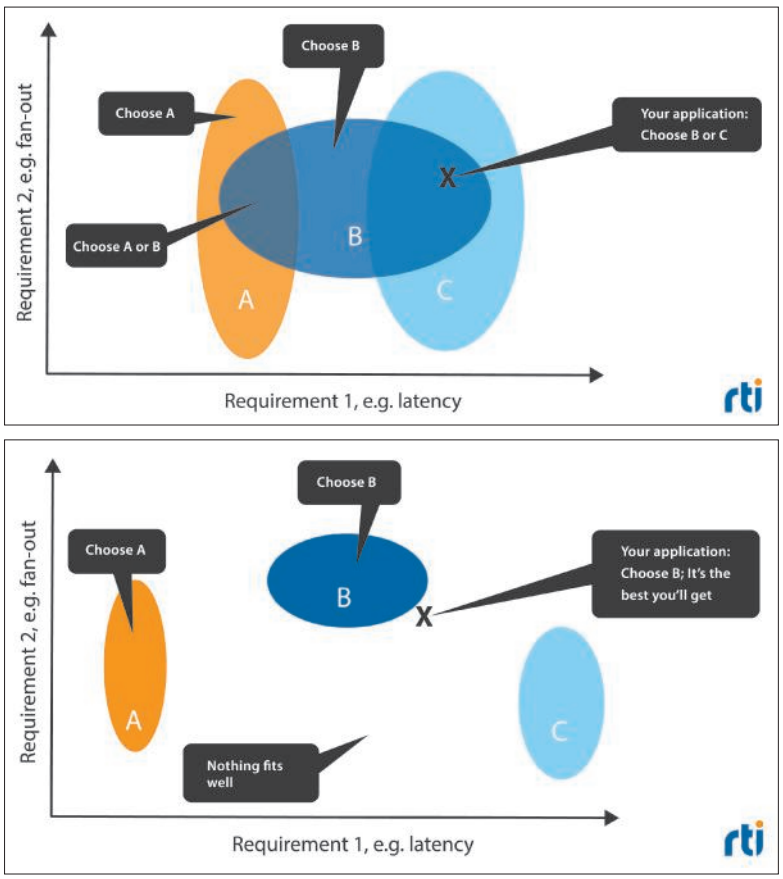
Data Distribution Service (DDS)

Here are five questions to answer and determine if you need DDS:

1. Is it a big problem if your system goes down for a short time?
2. Are milliseconds important in your communications?
3. Do you have more than 10 software engineers?
4. Are you sending data to many places, as opposed to just one?
5. Are you implementing a new IIoT architecture?

If you answered three out of the five questions "yes," you probably should use DDS.

DDS is a series of standards managed by the Object Management Group (OMG) that define a databus, which is data-centric information flow control. It's a similar concept to a database, which is data-centric information storage. The key difference: a database searches old information by relating properties of stored data. A databus finds future information by filtering properties of the incoming data.



Above: Before IICF, people assumed that competing standards met overlapping requirements. The designer's application in the top figure fits B or C. In reality, these standards do not overlap.

Both understand the data contents and let applications act directly on and through the data rather than with each other. Applications using a database or a databus do not have a direct relationship with peer applications.

The databus uses knowledge of the structure, contents and demands on data to manage dataflow. It can, for instance, resolve redundancy to support multiple sources, sinks and networks. The databus can control Quality of Service (QoS) like update rates, reliability and guaranteed notification of data liveliness. It can look at the data inside the updates and optimise how to send them or decide not to send them at all. It also can discover and secure data flows dynamically.

All of these things define interaction between software modules. The data-centric paradigm thus enables software integration.

OPC UA

OPC UA is a standard managed by the OPC Foundation (formally known as Object Linking and Embedding for Process Control) also documented as IEC 62541.

OPC UA targets device interoperability. Rather than accessing devices directly through proprietary application program interfaces (APIs), OPC UA defines standard APIs that allow changing device types or vendors. This also lets higher-level applications, such as human machine interfaces (HMI) find, connect to and control the various devices in a factory.

OPC UA divides system software into clients and servers. The servers usually reside on a device or higher-level Programmable Logic Controller (PLC). They provide a way to access the device through a standard "device model."

There are standard device models for dozens of types of devices from sensors to feedback controllers. Each manufacturer is responsible for providing the server that maps the generic device model to its particular device.

The servers expose a standardised object-oriented, remotely-callable API that implements the device model.

Clients can connect to a device and call functions using the generic device model. Thus, client software is independent of the actual device internals, and factory integrators are free to switch manufacturers or models as needed.

So, OPC UA can build and maintain a system from interchangeable parts, much like standardized printer drivers allow PC system integration. Note that the device model also provides a level of "semantic" interoperability, because the device model defines the generic object APIs in known units and specified reference points.

If you are in discrete manufacturing; building a device that will be integrated by control or process engineers or technicians, rather than software engineers; developing a product to be used in different applications in different systems, as opposed to one (type of) system where you control the architecture or building equipment for a "workcell" then the OPC UA is for you.

OneM2M

OneM2M provides a common service layer that sits between applications and connectivity transport. Its emphasis is on providing common services, on top of different connectivity standards.

To determine if you should use oneM2M, consider these questions:

1. Do you know what "ICT" stands for, and does it describe what you do?
2. Is the cellular network your primary connection technology?
3. Are your target applications largely composed of moving parts?
4. Can the components of the system tolerate intermittent connections and loosely-controlled latencies?
5. Will the system leverage services provided by a communications provider, such as a telco?

These questions differ in character from the questions about the previous technologies. OneM2M results from cooperation among many mobile wireless providers. It targets networks of mobile devices that communicate mostly or only through the base-station infrastructure.

RESTful HTTP

REST (Representational State Transfer) over HTTP is the most common interface between consumer applications and web services. REST is an architectural pattern for accessing and modifying an object or resource. One server usually controls the object; others request a "representation" and may then send requests to create, modify or delete the object.

To see if RESTful HTTP is the best candidate for your application, are you connecting independent devices to a single web service API; building an HMI interface to an IoT device or service; does your application only need to be fast enough for human interaction; must your dataflow cross firewalls that you do not control or is there no device-to-device communication?

Comparing these technologies highlights the stark differences and non-overlapping nature of connectivity approaches.

For instance, OPC UA is object oriented (OO), while DDS is data centric. Those are diametric opposites. The object-oriented mantra is "encapsulate data, expose methods."

Data centricity is all about exposing data, and there are no user-defined methods. The only methods are defined by the standard.

OPC UA targets final device-centric integration by plant engineers. It offers easy interoperability between devices from different vendors. By contrast, DDS targets final data-centric software integration by software teams. As intelligent software gains importance, DDS provides the global data abstraction and dataflow interface control that software teams need.

OneM2M and RESTful HTTP aim at connection from the edge to cloud services.

Looking at the differences, it's clear that these technologies simply do not compete in practice. However, the level of "confusional competition" is amazing. The various vendors and standards organisations, in general, do not help. Their positioning often uses similar words for vastly different concepts. Common terms like "publish subscribe" hide huge differences in types of information, discovery, selection of data and QoS control, and "Real time" without specifying a time period like milliseconds or minutes is meaningless.

Most applications fit one of these popular standards. If you answered "yes" to three questions for any technology above, you should start there. If you didn't, choose the closest match and start adapting.

Author details:
Dr. Stan Schneider is CEO at Real-Time Innovations

"Before the IICF, people assumed that competing standards met overlapping requirements in the IIoT space. It turns out they don't."
Dr. Stan Schneider