print | close

Overcome Protocol Challenges in Industrial IoT Environments

Electronic Design
Wenyuan Niu
Wed, 2016-09-07 10:13

A few years ago, the Industrial Internet of Things (IIoT) was little more than a buzzword. Today, it's a business reality in which communication and interaction capabilities can be extended to devices used for factory automation, renewable-energy applications, smart grids, intelligent transportation systems, and even oil and gas applications.

Yet field devices using different protocols can lead to daunting challenges in the implementation of an IIoT strategy. If devices can't speak in the same language, there's no data access. And without access to operational data, there's no IIoT.

Let's use the example of an automated processing plant. Intelligent field devices collect continuous, real-time data, including temperature, motor speed, start/stop status, and video footage, that's used to gain insights to increase competitiveness. But what if some of these devices use proprietary protocols, whereas others use open standard protocols? How do we manage this heterogeneous network?

Related

Wireless Sensor Networking for the Industrial IoT

Find Out About Industrial IoT at IoT Emerge

IoT—the Industrial Way

To do so, three steps need to be taken:

1. Selecting a suitable protocol.

2. Integrating different media and protocols.

3. Effectively handling proprietary protocols.

This article offers some hands-on advice to getting all three done on time and on budget. In the last step, we look at integrating legacy serial devices into the IIoT—a special challenge that warrants additional attention since most serial devices use proprietary protocols.

It can't be underestimated how important it is to overcome these challenges in order to implement successful Industrial IoT deployments, and how critical any technology is that can help reduce costs.

Select a Suitable Protocol

First, in order to solve the plants' pain points and allow management to successfully develop and deploy their ustrial IoT applications, the diverse devices must be able to share information and interact freely. The ıble is that field devices use different languages or protocols, with each of these protocols having its own unique qualities and features.

There exists an almost bewildering choice of connectivity options for engineers working on products and systems for the Internet of Things (IoT). In industrial environments, however, the two primary protocols are Modbus and EtherNet/IP.

The Modbus protocol is simple, easy to use, and cost-effective, making it ideal for collecting data every few seconds or even longer. Modbus is the lifeblood of a large number of automation applications. This de facto industrial communications protocol, following a master-slave architecture, has two major formats: Modbus RTU and Modbus TCP, a modified version of the former.

In essence, both formats fulfill the same functions. The main difference between them pertains to the physical layers they run on: Modbus RTU is used in serial communications (RS-232 or RS-485), while Modbus TCP facilitates communications over TCP/IP networks. Furthermore, Modbus TCP data packets use an additional header for TCP connections.

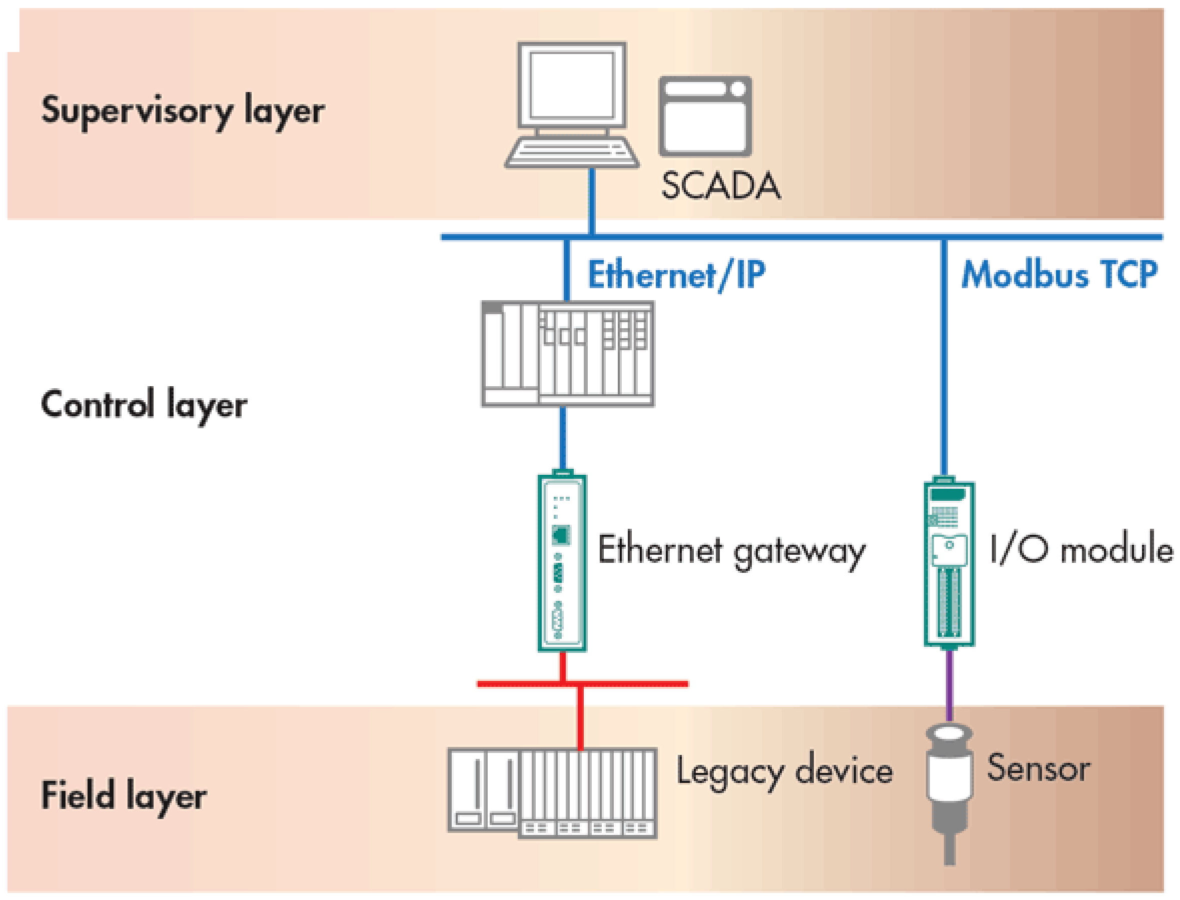
Need to exchange data much faster and more reliably? The EtherNet/IP protocol is your solution. It provides the network tools to deploy standard Ethernet technology—IEEE 802.3 combined with the TCP/IP Suite—for industrial automation applications while enabling internet and enterprise connectivity.

EtherNet/IP offers various topology options, including a conventional star with standard Ethernet infrastructure devices, or device level ring (DLR) with EtherNet/IP devices so enabled. QuickConnect functionality enables devices to be exchanged while the network is running. A flexible network architecture makes it compatible with copper, fiber, fiber ring, and wireless.

When upgrading a device or deploying a new IIoT application, it's critical that the field devices support open-standard protocols to ensure that they will be interoperable with devices from other vendors. Ethernet-based protocols are being used more frequently at field sites. Most are open-standard protocols, such as Modbus/TCP, EtherNet/IP, OPC UA in automation, or ONVIF in video.

Integrate Different Protocols and Media

A field layer, a control layer, and a supervisory layer typically comprise an IIoT network in the field, with each layer possessing assorted response times and environmental factors *(see figure)*.



Different protocols are deployed to meet the requirements of each of these three layers, adding to the need to develop communications between them. The solution? Deploy a protocol gateway. It lets you collect information from devices at each level, although they don't use the same protocol. For example, a protocol gateway can convert a field-layer protocol used in a Modbus RTU meter to a control-layer protocol such as EtherNet/IP.

It's critical that the right transmission media be used to build the most reliable IIoT communications system. Your vendor must provide a foolproof solution with regard to any communications issue, ranging from device interfaces such as RS-232/RS-485 or remote IO; different network interfaces like optical fiber, Ethernet, and wireless; and support for a variety of protocols whether it's fieldbus, OPC UA, or ONVIF.

Handling Proprietary Protocols

A large number of existing field devices—from power meters to barcode readers—use proprietary protocols. In most cases, their interface is serial.

To connect legacy serial devices to an Ethernet-based IIoT network, a serial-to-Ethernet converter, also called a serial-device server, is installed. Serial-device servers support two interfaces: a serial interface on the one side, and an Ethernet interface on the other side. Also, serial-device servers support virtual COM ports so that they work as legacy COM ports in an existing SCADA system, saving you the cost of developing a new one. Furthermore, serial-device servers support the raw socket mode, which transparently converts serial data to TCP or UDP packets.

Connecting a serial device to one SCADA system is simple and easy. When it comes to the IoT, however, it's not as straightforward, as the collected data may need to be transferred to the cloud. A SCADA system and a remote cloud application may issue a command simultaneously to a serial device via a serial-device server. Therefore, serial-device server needs to have a first in, first out (FIFO) queue to handle all queries. Only the first query be sent to a serial device, and the rest will wait in the FIFO queue.

Once the serial-device server receives the response from a serial device, it will send the response to the relevant SCADA system or cloud application and process the next query in the FIFO queue. This kind of command-by-command handling is very important in IoT multiple polling applications due to the large number of serial devices supporting proprietary protocols. Without this design, an extra IoT gateway that supports multiple polling is required.

A serial device server may not know how to divide serial data into TCP packets. Serial-device servers do not understand serial data formats, so they may put a response from a serial device into two or more TCP packets. This will result in a SCADA system or a cloud application rejecting these packets as incorrect responses, since they expect each TCP packet to represent a single response from a serial device.

To get around this issue, serial-device servers need to support flexible data-packaging options because the proprietary protocol may have a different format. For example, fixed data lengths or special delimiter characters can be used to identify single serial-device responses. This means a serial-device server will keep receiving data from a serial device and not transfer it to the Ethernet until it receives the fixed amount of data or a delimiter character. Without support for data-packaging options, you would have to develop complex SCADA software applications to handle the TCP packets properly. These alternatives waste time and may also create bugs in a system.

To handle large numbers of remote connections properly, serial-device servers should support flexible connection control. The best way to do this is to open a connection only when serial data is received from a device. When the transmission is completed, the serial-device server should close the connection as soon as possible. Without flexible connection control support, you will have to spend extra time handling connections at the central site or cloud application.

Conclusion

IIoT is far more demanding than either consumer IoT or traditional M2M. An Industrial IoT environment is comprised of legacy devices with older protocols and requires diverse means to communicate data with a vast spectrum of other connected devices on the network, ranging from sensors and robots to 3D printers and chemical mixing tanks. Communication is obviously a key component of the IIoT. Understanding how to overcome protocol challenges helps fulfill the promise of IoT for a more efficient, profitable manufacturing ecosystem.

Source URL: http://electronicdesign.com/iot/overcome-protocol-challenges-industrial-iot-environments