Protocolos IIoT para considerar

Por Aron Semle, Kepware - eFalcom, *www.efalcom.com*

Introducción

IoT es una sopa de letras: IIoT, IoE, HTTP, REST, JSON, MQTT, OPC UA, DDS, y la lista sigue. Conceptualmente, hemos discutido IoT (*Internet of Things*, 'Internet de las cosas') durante un largo tiempo y entendemos la idea básica y su viabilidad técnica. Ahora vayamos un poco más allá, identifiquemos casos de uso y construyamos prototipos. Es hora de meter cuchara en esta sopa.

Un gran desafío de IoT es la interoperatividad. En una encuesta reciente de *Nexus*, el setenta y siete por ciento (77%) de los entrevistados consideró que la interoperatividad es el mayor desafío de IoT.

Conectar dispositivos industriales con tecnologías de información y plataformas IoT es un tema considerable, y es de donde vienen un montón de abreviaturas. Existen varios protocolos para cumplimentar esto: algunos que son privados y otros que son estándares abiertos.

Todos están compitiendo para convertirse en protocolo único de IoT, pero es claro que eso nunca será una realidad. Estos protocolos coexistirán —cada uno con sus propias fortalezas y debilidades— y es nuestro trabajo entender dónde y cuándo usarlos.

Esta nota se centra en los estándares abiertos para conectar la industria con las tecnologías de la información, y casos de uso para cada uno.

Cliente/servidor vs. publicar/subscribir

A los fines de esta discusión, es importante agrupar los protocolos en dos categorías: cliente/servidor (*client/server*) y publicar/suscribir (*publish/subscribe*).

Los protocolos cliente/servidor requieren que el cliente se conecte al servidor y realice solicitudes. En este modelo, el servidor tiene los datos y responde a los pedidos del cliente, por ejemplo, el cliente quizá lee una temperatura, esto requiere que sepa acerca del servidor de antemano y sea capaz de conectarse.

Los protocolos publicar/suscribir requieren que los dispositivos se conecten a un "tópico" de un gestor intermediario y publiquen la información. Los consumidores se pueden conectar al gestor y suscribirse a los datos del tópico. Por ejemplo, un dispositivo puede medir la temperatura cada minuto y publicarlo una vez por hora. Una aplicación suscripta a dicha información recibirá una vez por hora un compendio horario de las muestras tomadas cada minuto. Este modelo desacopla al productor de datos del consumidor de datos.

Los protocolos cliente/servidor se utilizan mejor cuando uno conoce su propia infraestructura. Por ejemplo, uno sabe que su servidor en campo tiene una dirección IP (*Internet Protocol*, 'protocolo de Internet') de 55.55.55.55, en un puerto 1234. El cliente se puede conectar y realizar requerimientos.

Los protocolos publicar/suscribir son mejores cuando la infraestructura propia es desconocida. Por ejemplo, si un dispositivo remoto cambia de redes o tiene una conectividad intermitente, es más fácil para el dispositivo llamar a casa cuando se pone en línea y publicar su información.

En términos de pros y contras, los protocolos cliente/servidor son más compatibles y seguros porque están basados en conexiones punto a punto. Sin embargo, son menos escalables porque las conexiones punto a punto son más difíciles de manejar y mayor demandantes de recursos.

Por el contrario, los protocolos publicar/suscribir son más escalables porque el separar a productores de consumidores permite que cada uno se agregue y se quite de forma independiente. Por consiguiente,

asegurar estos protocolos es más complejo porque involucran más piezas. También pueden aparecer cuestiones de compatibilidad dada la separación entre productor y consumidor. Por ejemplo, cambiar el formato del mensaje que envía el productor requiere que todos los consumidores se adapten al nuevo formato.

Ahora que entendemos las categorías básicas, observemos con un poco más de detalle a los protocolos cliente/servidor y publicar/suscribir.

Protocolos

Los protocolos que discutiremos tienen el potencial de conectar dispositivos industriales con plataformas IoT. Quizá no haga falta aclararlo, pero si usted está tratando de conectar dos aplicaciones y las dos soportan HTTP, pruebe con HTTP primero. Si eso no funciona o si el medio no lo tolera, siga leyendo. Describiremos cada protocolo y cuándo usarlos. A continuación, una breve lista de lo que trataremos:

- OPC UA
- HTTP (REST/JSON)
- MQTT
- CoAP
- DDS
- AMQP

OPC UA

OPC UA (*Unified Architecture*, 'arquitectura unificada') es el estándar de nueva generación que le sigue a *OPC Foundation*. OPC clásico es bien conocido en la industria y provee una interfaz estándar para comunicarse con los PLC (*Programmable Logic Controller*, 'controlador lógico programable'). OPC UA pretende expandir la compatibilidad de OPC al nivel de los dispositivos y de las empresas.

OPC UA es un protocolo cliente/servidor. Los clientes se conectan, navegan, leen y escriben al equipamiento industrial. UA define la comunicación desde la aplicación hacia la capa de transporte, lo que lo hace muy compatible entre vendedores. También es muy seguro, y usa mensajes bidireccionales firmados y encriptación de transporte

OPC UA tiene una amplia base instalada en el mundo industrial. Es una buena solución para conectar información de sensores y PLC en aplicaciones industriales ya existentes como sistemas MES (*Manufacturing Execution System*, 'sistema de ejecución de manufactura') y SCADA (*Supervisry Control and Data Acquisition*, 'supervisión, control y adquisición de datos'), en donde la conectividad OPC y OPC UA ya estén disponibles.

Sin embargo, OPC UA es nuevo para las tecnologías de información. Algunas personal de TIC (tecnologías de la información y comunicación) se asustan ante la complejidad de UA en comparación con otros protocolos TIC. Bastante de esta complejidad reside en el hecho de que OPC UA sea un protocolo industrial, pero esta percepción ha llevado a ralentizar su adopción para plataformas IoT y la comunidad de código abierto.

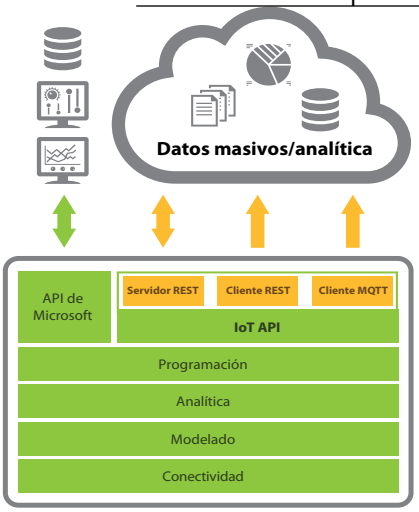
Pero la cosa están cambiando: hace muy poco, *OPC Foundation* abrió el código del estándar OPC UA para hacerlo más accesible y colaborar a que se incremente su adopción.

Por ahora, use OPC UA cuando necesite información del sensor y de PLC dentro de las soluciones MES y SCADA ya existentes, y esté atento a la adopción que los proveedores de plataformas IoT y la comunidad de código abierto hagan de OPC UA.

HTTP (REST/JSON)

HTTP (*Hypertext Transfer Protocol*, 'protocolo de transferencia de hipertexto') es un protocolo cliente/servidor sin conexión ubicuo en TIC y en la web. Dado que existen incontables herramientas de código abierto que usan HTTP, y que todo lenguaje de codificación tiene bibliotecas HTTP, es muy accesible.

El foco de HTTP en IoT gira en torno a REST (*Representational State Transfer*, 'transferencia de estado representacional'), que es un modelo sin estados previos donde los clientes pueden acceder a



recursos en el servidor a través de pedidos. En la mayoría de los casos, un recurso es un dispositivo y la información que tal dispositivo contiene.

HTTP provee transporte, pero no define la presentación de la información. Así, un requerimiento HTTP puede contener HTML, JavaScript, JSON (*JavaScript Object Notation*, 'notación de objeto JavaScript'), XML, y demás. En la mayoría de los casos, IoT está estandarizando JSON para HTTP. JSON es similar a XML pero sin la sobrecarga ni esquema de validación por lo que es más liviano y flexible. JSON también es soportado por la mayoría de las herramientas y lenguajes de programación.

La industria cuenta con algo de experiencia usando HTTP para la configuración de productos y dispositivos, pero no para el acceso a datos. De este modo, muchas plataformas TIC e IoT aceptan HTTP para proveer y recibir información, pero no así las plataformas industriales. Esto está cambiando a medida que cada vez más puertos y PLC agregan HTTP nativo.

Use HTTP para enviar grandes cantidades de información, como lecturas de temperatura minuto a minuto cada hora. No use HTTP para información de video de alta velocidad. HTTP puede operar bajo el segundo, pero actualizaciones de cien milisegundos (100 ms) con HTTP son difíciles. Implica bastante sobrecarga por mensaje, así que enviar mensajes pequeños es ineficiente. Y siempre asegure la comunicación con HTTPS. La sobrecarga es mínima.

Esté atento a las cuestiones de interoperatividad con los productos HTTP. Solo porque dos productos soporten HTTP, REST y JSON no significa que todos estén listos para usar. Muy a menudo, los formatos JSON son diferentes y requieren de una mínima integración para que las cosas funcionen.

MQTT

MQTT (*Message Queuing Telemetry Transport*, 'Cola de mensajes telemetría y transporte') es un protocolo publicar/suscribir diseñado para SCADA y redes remotas. Se centra en un mínimo encabezado (dos bytes de cabeza) y comunicaciones confiables. También es muy simple. Tal como HTTP, la carga MQTT es específica para la aplicación, y la mayoría de las implementaciones usan un formato JSON personalizado o binario.

MQTT no es tan ampliamente utilizado como HTTP, pero aún tiene una gran participación en el mercado de TIC. Existen muchos ejemplos, proyectos, clientes/productores de código abierto en cada lenguaje. Muchas plataformas IoT soportan HTTP y MQTT como los primeros dos protocolos de entrada de información.

Use MQTT cuando el ancho de banda sea premium y no conozca su infraestructura. Asegúrese de que su vendedor tenga un gestor MQTT a quien le pueda publicar información, y siempre asegure la comunicación con TLS (*Transport Layer Security*, 'seguridad en la capa de transporte').

¿La aplicación final no soporta MQTT? Si la respuesta es "no", existen varias herramientas de código abierto para incluir información de MQTT en las bases de datos y otros formatos como HTTP.

Esté atento a cuestiones de compatibilidad similares a HTTP. Que dos aplicaciones soporten MQTT no quiere decir que puedan operar entre sí. El tópico y los formatos JSON quizá necesiten ajustarse para que dos productos puedan operar entre sí.

CoAP

CoAP (*Constrained Application Protocol*, 'protocolo de aplicación restringida') fue creado por IETF (*Internet Engineering Task Force*, 'Grupo de Trabajo de Ingeniería de Internet') para proveer la compatibilidad de HTTP con una mínima carga. CoAP es similar a HTTP, pero usa UDP/multicast en lugar de TCP.

Además, simplifica el encabezado HTTP y reduce el tamaño de cada requerimiento.

CoAP se utiliza en dispositivo de borde en donde HTTP sería demandante de recursos, y a menudo, las plataformas de IoT lo utilizan como tercer protocolo, después de HTTP y MQTT. Similar a HTTPS, CoAP usa DTLS (*Datagram Transport Layer Security*, 'seguridad en la capa de transporte datagrama') para proteger las comunicaciones.

Use CoAP cuando HTTP demande un ancho de banda demasiado intenso. Recuerde que su adopción en el mercado no está tan extendida como HTTP, de modo que quizá limita sus opciones de software y hardware. Existen soluciones para convertir mensajes CoAP desde y hacia HTTP que pueden hacer a las soluciones CoAP más interoperables.

DDS

DDS (*Data Distribution Service*, 'servicio de distribución de datos') es un protocolo publicar/suscribir que se focaliza en el borde de la comunicación en la red. DDS es un estándar abierto operado por OMG (*Object Management Group*, 'Grupo de Gestión de Objetos'). A diferencia de MQTT, que requiere de un agente centralizado, DDS está descentralizado. Los nodos de DDS se comunican directamente punto a punto a través de UDP/multidifusión (multicast). Esto hace que no sea necesaria una gestión centralizada de la red y que DDS sea un protocolo más veloz, con una resolución por debajo del milisegundo.

DDS es una buena solución para la entrega de información de forma confiable y en tiempo real. Úselo para comunicaciones rápidas M2M (*machine to machine*, 'máquina a máquina').

DDS soporta a los gestores para integrar redes DDS con la empresa, pero en la práctica no está bien posicionado como punto de integración entre la industria y TIC; como gestores, son a menudo secundarios para la red DDS.

AMQP

AMQP (*Advanced Message Queuing Protocol*) es otro protocolo tipo publicar/suscribir que proviene del sector de servicios financieros. Tiene su presencia en TIC, pero bastante limitada en la industria.

El mayor beneficio de AMQP es su modelo robusto de comunicaciones que soporta transacciones. A diferencia de MQTT, AMQP puede garantizar transacciones completas —lo cual, aunque útil, no siempre es algo que requieran la aplicaciones IoT—.

AMQP se agrupa a menudo con protocolos IoT y es uno, pero su mayor contra es que se trata de un protocolo pesado. Fue destinado par sistemas TIC, y no para el límite de la red.

Conclusión

OPC UA, HTTP, MQTT, CoAP, DDS, y AMQP todos tienen lugar en IoT. Cuál de estos protocolos tiene la mayor parte del mercado no está claro, pero cada uno tiene sus pros y sus contras. Es importante elegir el protocolo que mejor se adapte a sus necesidades, y seleccionar los socios tecnológicos que se puedan adaptar a tales protocolos. Esto asegurará el éxito para sus aplicaciones IoT y lo protegerá de las guerras de protocolos.

Asegúrese de estar al tanto del nuevo *gateway* (puerto) IoT de *Kepware* disponible en el lanzamiento de *KEPServerEX* versión 5.19. Incluye soporte para REST y MQTT, permitiendo a los clientes introducir la información de PLC en las nuevas plataformas IoT y herramientas de código abierto como Node-RED. ❖

Nota del editor: *Kepware* es una empresa estadounidense dedicada a facilitar la conectividad entre dispositivos industriales y la empresa. En Argentina, disponible a través de la representación de *eFalcom*.

Nota del editor: La nota aquí reproducida fue originalmente escrita como nota técnica de la empresa *Kepware* para su publicación en *Kepware Whitepaper*.