

SHARE  

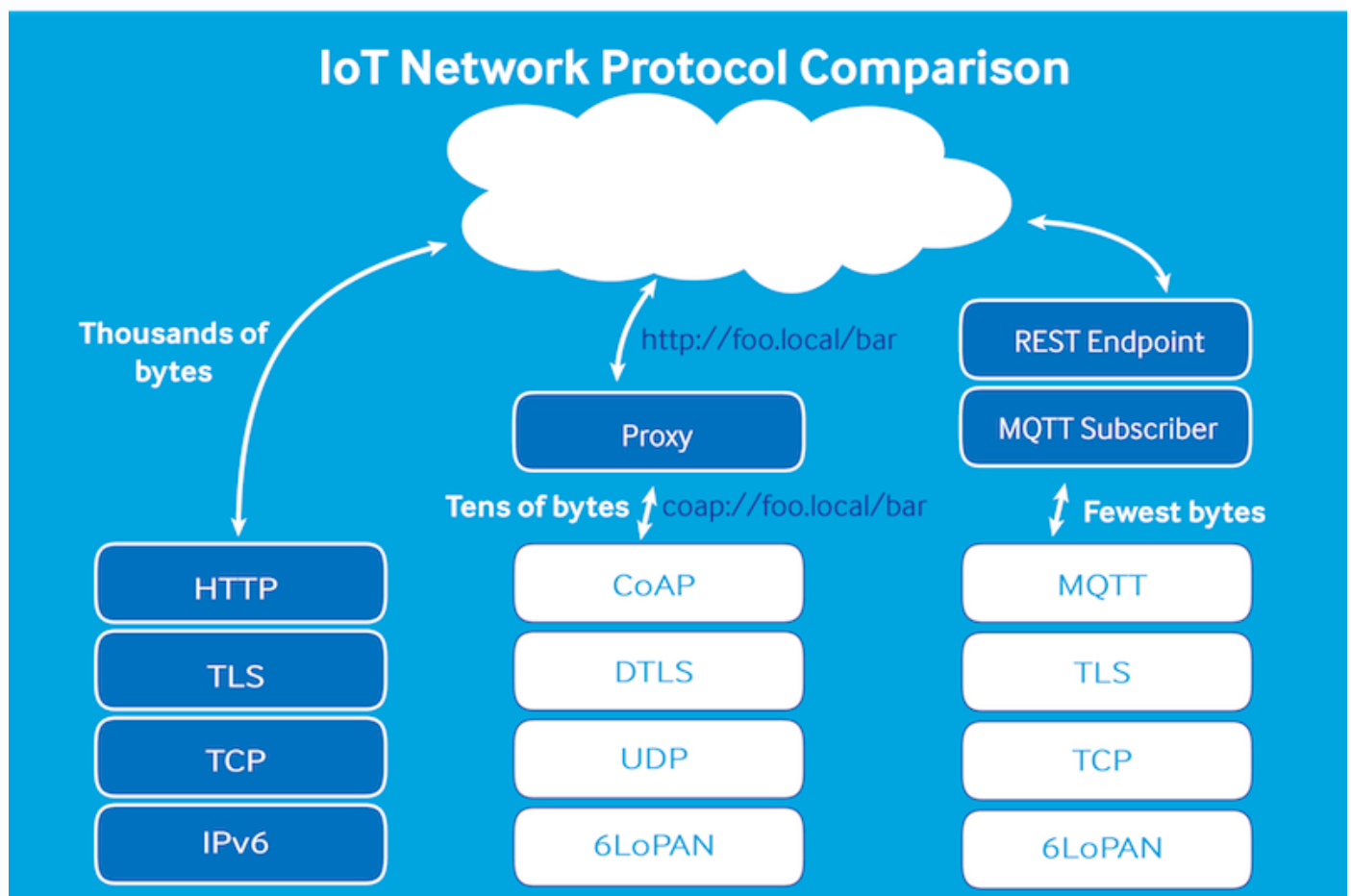
IoT 101: Networks

KEVIN SHARP • 16 SEP 2015

In this installment of our IoT 101 series we discuss how to select network protocols based on the way you want to interact with your IoT. Everything we'll cover applies to developers of IoT devices and apps, whether or not you're working with ARTIK.

When selecting a networking protocol for your IoT system, it helps to think about the "thing" you want to interact with. Is each node in your IoT an entity with which you want to interact directly, or do you want to monitor and control a system that's only useful as an aggregate of IoT nodes?

Let's consider a renewable energy production field in a windy, sunny location. Wind turbines perform their air ballet 30 meters above thousands of solar panels; a transmission line connects the field to the power grid. In this scenario I would want to interact directly with each turbine, treat the panels in groups, and monitor the power line over a satellite link.



The CoAP protocol offers a REST programming model easy to proxy to web applications, while MQTT requires the least bandwidth from IoT nodes.

When every byte counts

The MQ Telemetry Transport (MQTT) network protocol would be a good fit for monitoring and controlling the solar panels. MQTT is a publish/subscribe protocol with central message brokers. Each solar panel could contain an IoT node publishing voltage, current, and temperature messages. Each rack of solar panels would include an IoT node with a message broker that relays status messages to all subscribers. It would be simple to aggregate the data streams from individual panels into a stream of arrays containing data from all panels in the rack.

MQTT is designed to minimize bandwidth, which makes it a good choice for satellite monitoring of the transmission line, but there's a catch. No metadata in message headers means interpretation of messages is completely up to the system designer.

The easiest way around this problem is by careful consideration of how you publish messages. MQTT messages are published to addresses called Topics, and Topics can be hierarchical. For example, an IoT node could publish to the topics **aisle15/rack20/panel5/temp** and **aisle15/rack20/panel5/volts**. A subscriber can use wildcards to monitor an entire rack: **aisle15/rack20/+/temp**.

Real world networks

To compensate for unreliable networks, MQTT supports three Quality of Service (QoS) levels: Fire and Forget (0), at least once (1), and exactly once (2). If QoS level 1 or 2 is requested, the protocol handles message retransmission to guarantee delivery. QoS can be specified by publishing clients (covers transmission from publisher to broker) and subscribing clients (covers transmission from broker to subscriber).



Tip: MQTT QoS 2 will increase latency because each message requires two full round-trip handshakes from sender to receiver (four total from publisher to subscriber).

Last Will and Testament

In a publish/subscribe pattern it's difficult to know the difference between "It's been a while since I heard from my publisher" and "My publisher died". That's where the MQTT Last Will and Testament comes in. Clients can publish messages to dedicated topics (e.g., **aisle15/rack20/panel5/sensorfault**) to be delivered if they disconnect without sending a graceful "disconnect" message. The messages are stored on the broker and delivered to anyone who subscribed to the Topic.

MQTT at a glance

- Very low bandwidth
- TCP/IP
- Publish/subscribe message transfer

- Many-to-many topology through central broker
- No metadata
- Three QoS levels
- Last Will and Testament reveals disconnected nodes

The RESTful option

For the wind turbine IoT nodes I'd prefer a more API-based interface. The Constrained Application Protocol (CoAP) uses the familiar REST design pattern where servers make resources available at a URI and clients access the resources using methods including GET, PUT, POST and DELETE. Published standards makes it easy to interpret **content formats**: for example, XML (ID=41) or JSON (ID=50).

A URI to access speed of one of the turbines might be: **coap://turbine20.domain.tld/speed** .

Turbines could query each other to optimize performance of the array. For example, a turbine could query the adjacent upwind turbine to determine if wind was increasing or decreasing. The neighbor could in turn query its upwind neighbor, and so on. This simple rule could allow a turbine to anticipate wind changes and prepare accordingly. If the entire array has access to the web (via HTTP) and a DNS server, it's easy to translate the URI to: `http:// turbine20.domain.tld/speed` and monitor the field remotely.

Where nobody knows your name

Even if we don't have a full DNS server available to our turbines we can still address them by host name as long as all network nodes support multicast DNS (mDNS) (available on **ARTIK 5** and **ARTIK 10** chips). Here's how it works:

- Query to **coap://turbine20.domain.local/speed** causes mDNS to issue a multicast message to all nodes on the local network.
- Host **turbine20.domain.local** replies with IP address in a multicast message to all nodes.
- Local mDNS service caches result.



Tip: It is possible but not recommended to allow both mDNS and (standard) DNS name resolution on the same subnet. Best practice would be to support mDNS for communication among CoAP nodes, with an HTTP-CoAP proxy.

CoAP at a glance

- REST client/server document transfer
- Easy to translate to HTTP for web integration
- One-to-one topology with direct connections
- Metadata to differentiate document types

- UDP
- Security via DTLS

Other networking features

ARTIK 5 and ARTIK 10 chips also support the following network features:

- **LWM2M** is a device management protocol implemented with CoAP.
- **IPv6** is the addressing scheme that allows the extension of IP addressing to IoT networks.
- **6LoWPAN** is the implementation of IPv6 over **low power Wireless Personal Area Networks** (802.15.4)

Explore the IoT 101 series: *Connectivity*

About the author: *Kevin Sharp* has been an engineer since long before he got his engineering degree, and has extensive experience in data acquisition and control networks in industrial, retail, and supply chain environments. He's currently a *freelance writer* based in Tucson, Arizona.

Recent Posts

ARTIK GPIO: Using Digital Outputs IoT 101: Networks

Make An IoT Weather Station With ARTIK, Temboo And SAMI IoT 101: Connectivity





Partner Profile: SIGFOX SAMI + ARTIK = Secure IoT ARTIK Is On The Road To Beta

Around The World With ARTIK Samsung IoT: 2020 And Beyond Partner Profile: Medium One

ARTIK

Home Media
Inspiration Developer
Partners Blog
Hardware Documentation

CONNECT

 Twitter
Instructables
 Facebook
 LinkedIn
 Instagram

RELATED SITES

SAMI
Voice of the Body
SSIC



Feedback



Join Mailing List

