print | close

# Open Protocols Expand M2M's Boundaries

*Electronic Design*

Benjamin Cabé
Benjamin Cabe, Contributing Technical Expert
Thu, 2013-10-24 12:25

Machine-to-machine (M2M) communications present a unique set of technical challenges that any connected application must address. M2M applications must be able to connect over networks that may be unreliable, and that often have limited bandwidth, especially when connecting over cellular networks. M2M devices themselves also often have limited processing power and a need to minimize power consumption.

**Related Articles**

- M2M IWG Enables M2M Development With Open-Source Technologies
- Use Lua To Develop M2M Embedded Applications
- M2M With The Intelligent Device Platform

As a result, connected applications require M2M-specific capabilities and considerations in their core communication mechanisms. In many cases, however, the mechanisms for performing these functions have been closed and proprietary.

Often, a developer will choose a specific M2M module, and that module comes with its own software development kit (SDK), usually with proprietary mechanisms for device and application management. As a result, developers and OEMs become effectively locked into that hardware—as well as that vendor's preferred server and management framework—limiting flexibility and interoperability.

Increasingly, developers, OEMs, and customers are demanding more open and interoperable M2M protocols. Open protocols provide a number of advantages, enabling developers to:

• Fully view and understand device and application management mechanisms

• Tweak and adapt those mechanisms as needed for a given application

• Reuse the same approach across different hardware and M2M service platforms

Open protocols also capitalize on a global community of M2M developers who are testing and vetting these mechanisms, as well as continually expanding their applicability by implementing them in new programming languages.

For all of these good reasons, the global M2M community has been working to develop and expand open protocols for M2M applications. Sierra Wireless is working closely with the Eclipse Foundation and other M2M industry leaders as part of the M2M Working Group to lead the way in many of these efforts. Other industry groups such as the Open Mobile Alliance (OMA) are hard at work to expand open M2M protocols as well.

These protocols fall into two categories:

• Device management protocols that provide mechanisms to perform basic management functions such as monitoring the status of an M2M module, monitoring signal strength, encrypting communication, and performing firmware upgrades

• Protocols for managing the "business data" of the M2M application, for example, sending temperature readings from a sensor to a back-end server

Let's take a closer look at these efforts.

**Increasing Efficiency**

One of the earliest and most important efforts around device management was the OMA Device Management (OMA DM) protocol. OMA DM has been a hugely successful standard for connected device management, and it is used in millions of devices in the field today. (In fact, one of the most popular elements of the Koneki Project—the set of open-source M2M tools being developed by the M2M working group—is an OMA DM simulator.[1]) But OMA DM has some limitations.

First, the standard itself is extremely large and complex. Also, because OMA DM is based on standard Hypertext Transport Protocol (HTTP), it is not optimally bandwidth-efficient. This is especially true for applications that require secure communications, which have historically required HTTP Secure (HTTPS). For an embedded module, HTTPS is extremely costly in terms of bandwidth consumed and roundtrips required between client and server for authentication. For some M2M applications—for example, low-power remote or mobile devices that require secure connectivity—HTTPS is simply not viable.

That's why the OMA has been working with industry partners for several years to develop the next generation of OMA DM: Lightweight M2M (LWM2M), which is expected to be standardized by the end of this year.

The most significant change in LWM2M is a shift from HTTP to a newer standardized transport mechanism, Constrained Application Protocol (CoAP). Designed specifically for low-power Internet-connected devices, CoAP replicates the HTTP paradigm, along with simplified Representational State Transfer (RESTful) architecture mapping, but over the much more bandwidth-efficient User Datagram Protocol (UDP).

CoAP even allows devices to communicate over Short Message Service (SMS). And, CoAP and LWM2M can support secure encrypted communications without HTTPS. With these capabilities, LWM2M can provide an extremely efficient, secure, open, and interoperable mechanism for M2M device management. As a result, it will likely be supported by virtually all M2M hardware in the coming years.

**Bringing Publish/Subscribe Paradigms To M2M**

Another important open M2M protocol is Message Queue Telemetry Transport (MQTT), which provides an extremely lightweight publish/subscribe messaging transport model for M2M applications. Effectively, it allows various clients (for example, an M2M gateway) to communicate via a central broker.

Clients communicate by "subscribing" to the central broker to receive specific notifications to which the client has subscribed and "publishing" their own messages to the broker. Because of these capabilities and its small code footprint, MQTT can provide an extremely efficient one-to-one or one-to-many communication method for M2M devices connecting over constrained networks.

While MQTT has been a de facto standard for many years, the Organization for the Advancement of Structured Information Standards (OASIS) is currently working to officially standardize it. The M2M Working Group is also helping

to expand the applicability of MQTT to new programming languages and users as part of its Paho project. One of the primary elements of the group's efforts to expand open-source M2M, the Paho project is creating reference implementations of MQTT in C/C++, Java, Python, Lua, Objective C, and other languages, all of which is helping MQTT gain significant traction in the industry.

The Eclipse M2M effort will also soon host an open-source implementation of an MQTT broker. Not only will developers have access to reference implementations for clients, they also will be able to take the MQTT broker source code and run it as part of their own server infrastructure. Ultimately, developers will have access to full open-source MQTT across the end-to-end communication chain.

**Optimizing Application Data Payloads**

In addition to M2M device management enhancements, M2M applications can benefit from optimizations to the business data payloads in M2M communications. Another open protocol, Micro M2M Data Access (M3DA), addresses this need. Developed by Sierra Wireless and contributed to the Eclipse effort as part of the Mihini project (an open-source application framework based on Sierra Wireless technology), M3DA provides a binary serialization mechanism designed to optimize bandwidth consumption of M2M payload data.

Consider a home automation application where a sensor acquires a temperature reading of a room every minute and reports that data back to a server once per hour. For an application running over a 2G GSM cellular network, even reporting this relatively small amount of data can consume a significant amount of bandwidth. Now, imagine that the temperature of the room stays at 70°F for the entire hour. A traditional M2M application would report that value 60 times. With M3DA, the sensor can instead report the equivalent of 70°F x 60.

In this way, M3DA can significantly optimize M2M payloads. M3DA can also be used in conjunction with other M2M-optimized transport protocols such as MQTT.

**Bridging The Gap Between M2M Protocols**

Along these lines, it's important to recognize that none of these protocols is the single "right" choice for M2M communications. There will always be a range of protocols available to M2M application developers, and different protocols (or combinations of protocols) will be better suited to different applications.

The M2M Working Group's proposed Ponte project (ponte is Italian for "bridge.") aims to make it easier for applications to integrate M2M communications into other types of applications using a common application programming interface (API). Ideally, developers will be able to, for example, use gateways and sensor devices using MQTT or CoAP, send those communications to Ponte, and convert them so they can integrate with REST/HTTP systems.

The project holds the potential to break down the wall that exists between the M2M and HTTP worlds. But as Ponte is a fully open-source project, it could become much more, allowing developers to create extensions to translate any M2M protocol to any other.

**Unlocking New Possibilities**

These developments are extremely exciting for those of us in the open-source community. But they should be even more exciting for the larger M2M community and the technology industry as a whole.

The fact that growing numbers of developers, hobbyists, and open hardware communities are adopting open-source M2M

protocols suggests that these protocols are practical and easy to use. Broad adoption will create lasting benefits. As users build on basic M2M building blocks, they will add to them, gain and share expertise, develop best practices, and ideally develop their own open-source M2M code that others can draw on. Ultimately, this openness and interoperability will expand the boundaries of M2M and unleash a new generation of applications that capitalize on the Internet of Things.

Reference

"Koneki Open-Source Development Tools Simplify M2M Development," Benjamin Cabé, http://electronicdesign.com/communications/koneki-open-source-development-tools-simplify-m2m-development

***Benjamin Cabé*** *is an open-source evangelist at Sierra Wireless. He has a longtime passion for Eclipse and its ecosystem and is a committer on several Eclipse projects and contributor to numerous other open source projects. In his day-to-day job, and as the chair of the Eclipse M2M Industry Working Group, he advocates the use of innovative technologies for the Internet of Things. He also co-leads Eclipse projects Koneki and Mihini. When not wandering on the Internets engaging M2M/IoT developers, he is building crazy machines with Arduino kits and Raspberry Pis. You can find him online on Twitter (@kartben) or on his blog, http://blog.benjamin-cabe.com. He can be reached at* bcabe@sierrawireless.com.

**Source URL:** http://electronicdesign.com/embedded/open-protocols-expand-m2m-s-boundaries