EPRI ELECTRIC POWER RESEARCH INSTITUTE

LIGHTWEIGHT MESSAGING TECHNOLOGIES FOR THE ENERGY INTERNET OF THINGS: AN INTRODUCTION

Abstract

Current wide-area and local networks based on the familiar Internet protocol suite ("TCP/IP") are in widespread use by utilities and utility customers. As the grid evolves to include ever-greater levels of automation, and as the use of communications-capable edge devices expands, some limitations of these protocols (and the applications built upon them) are becoming apparent. Widespread use of traditional protocols with a lower-capability edge device is often limited by the device's constrained processing and memory capacity or its connection via lower-speed networks. This paper presents options for reducing network complexity and message sizes and introduces examples of alternative application-layer messaging systems that leverage traditional Internet-based networks while reducing the burden on smaller devices at the grid edge.

KEYWORDS

Internet Protocols
TCP/IP
Messaging Systems
Constrained Applications
Internet of Things (IoT)

Introduction

The central role of communications in the integrated energy grid is well-established. Current wide-area and local networks based on the familiar Internet protocol suite ("TCP/IP") are in widespread use by utilities and utility customers. Successful distribution automation, whether in centralized or decentralized form, requires timely communication with assets located toward the edge of the grid. Management of loads and customer-sited generation can reduce stress on the grid at peak times, may allow deferral of costly grid upgrades, and can support controlled expansion of distributed generation. And all of this requires practical communications solutions in order to work.

As the grid evolves to include ever-greater levels of automation, and as the use of communications-capable edge devices expands, some limitations of familiar protocols (and the applications built upon them) are becoming apparent. This

Table of Contents

evolving environment is often called the "Energy Internet of Things," alluding to its focus on communications between physical devices, including vehicles, home appliances, and sensors or actuators with embedded networking capabilities. This represents a dramatic extension of Internet connectivity beyond the human-interface devices such as PCs, laptops, smartphones, and tablets.

This emerging world of machine-to-machine networking is not unfamiliar to utilities: after all, the smart grid itself is energy IoT in action, leveraging as it does such technologies as Advanced Metering Infrastructure (AMI) and distribution automation. However, the next wave of energy IoT communication requirements is being driven by the emergence of a plethora of smaller Internet-connected load devices, such as switches, power outlets, lightbulbs, televisions, and sensors. These devices, remotely accessible by users, aggregators, or utilities, are being announced frequently by manufacturers and their deployment is growing: 2017 saw a 31% global growth in the number of devices[1] and a near-quadrupling of devices by 2020 has been forecast.[2]

Widespread use of these smaller devices for energy IoT is often limited by their constrained processing or memory capabilities or their connection via lower-speed networks. Furthermore, these devices are often price-sensitive commodity products that cannot remain competitive if they adopt more expensive hardware. Nevertheless, there is continuing interest in adding communications capabilities to ever-smaller devices.

Therefore, it is not surprising that recent experience with implementing some emerging IoT use cases using such edge devices has revealed issues with using existing communications technologies. Therefore, there is continual pressure to implement communications technologies that require less processing power, less memory, and less network bandwidth.

Avenues of Approach

While the issues described above can be addressed via several paths, the extensive investments utilities have made in Internet protocol technologies suggest that we avoid immediate consideration of replacing these proven and widely supported networks. This paper will therefore primarily focus on options and alternatives at the Application Layer of a traditional network architecture (see sidebar), leaving aside considerations of alternative technologies applicable to lower levels of the network.

Reducing Network Complexity

Although the primary focus of this paper is on protocols and technologies at the Application layer of the network, options for bandwidth or power reductions based on standard options within Internet protocol suite itself should not be ignored. The Transport layer offers some options that may be useful with IoT devices.

TCP communication establishes a session between the source and destination nodes. This has security advantages and provides guarantees of correct message delivery that are important for many information exchanges, including those that are critical for grid reliability or that have significant financial implications for stakeholders.

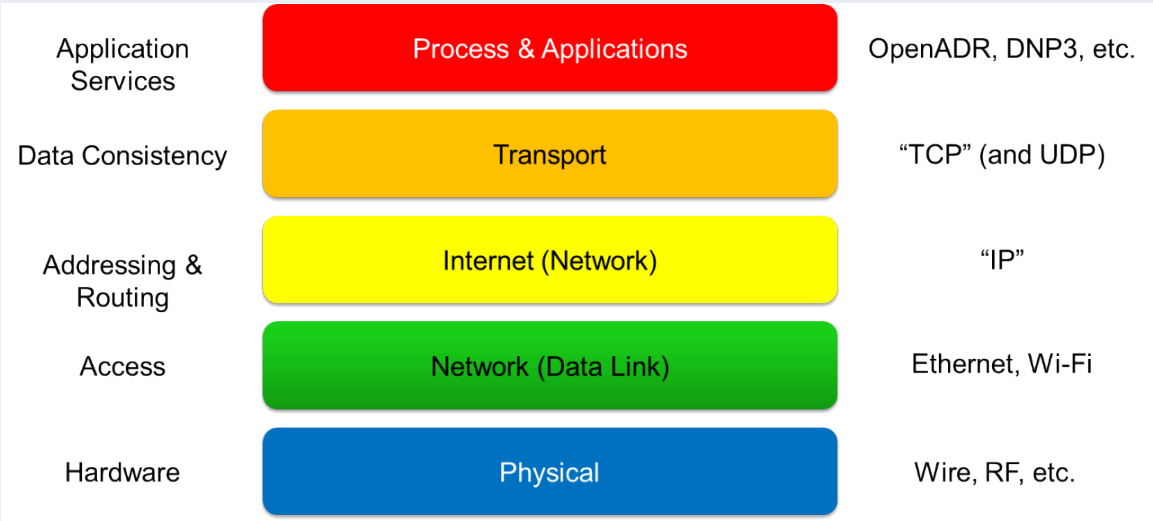
Nevertheless, not all information in the integrated grid requires these features and the network can be configured to avoid the processing overhead and network bandwidth required by TCP. A less-reliable method of information exchange is offered by the *User Datagram Protocol* (UDP), which is a standard part of all Internet protocol suites. Relative to IP, UDP provides very little in the way of additional services: it does not guarantee delivery of message packets, nor can it create secure sessions between nodes. About all it does is offer a simple checksum-based error detection service so that corrupted messages can be detected. However, it asks much less of both processing and network bandwidth.

Applications that can make effective use of such a streamlined protocol are those that send large amounts of (nearly) redundant data, where recovery of a missing packet every so often is not worth the processing and bandwidth cost of using TCP. Classic examples of the use of UDP include streaming media (such as audio, video, or "Voice over IP"). For utilities, telemetry from IoT sensors that can stream data points from throughout the grid would be a natural application for UDP. In such applications, high-speed data streams can afford to drop an occasional data point with little impact to the system.

1 R. Köhn, "Konzerne verbünden sich gegen Hacker," *Frankfurter Allgemeine*, 16 February 2018.

2 A. Nordrum, "Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated," *IEEE Spectrum*, 18 August 2016.

Layered Network Architecture Terminology



A layered network architecture

In a layered architecture, a hierarchy of conceptual layers is combined to provide the supported network services. Each horizontal layer leverages and supplements the services provided by the layer immediately beneath it to offer a set of higher-level services to the layer immediately above it. Each layer uses its appropriate network protocols to exchange information with its peer instance on another node in the network. At the Physical layer, some medium (wires, optical fiber, radio signals, etc.) carries the messages, while the higher layers support the application function at the highest level (application protocols like OpenADR or DNP$_3$ and supporting protocols, such as the Hypertext Transport Protocol used by the World Wide Web).

In the Internet protocol suite, the **Network or Data Link** *layer is the lowest and isolates the higher-level services from the underlying physical medium. This layer is what allows the Internet to operate over Ethernet, Wi-Fi, cellular data networks, and all the others without affecting the higher layers in the stack.*

The **Internet** *layer (confusingly sometimes referred to as the* **Network** *layer) provides various control and management services, including forwarding messages from one node to another. It uses addresses in the* **Internet Protocol** *(IP) to route messages toward their destinations. However, IP is only aware of individual hops between adjacent nodes: it knows nothing about preceding or subsequent hops needed to reach the eventual recipient. Since it cannot guarantee that every packet of information follows the same path between nodes, some packets may be lost in transit, duplicated, or arrive out of sequence.*

It is the job of the **Transport** *layer to correct whatever shortcomings of the Internet layer may be desired. One of the main protocols at this layer is the* **Transmission Control Protocol** *(TCP), which establishes reliable end-to-end connections between the source and destination nodes. TCP requests retransmission of missing packets, eliminates duplicates, and establishes a "session" or "connection" between the source and destination nodes that provide a reliable and (optionally) secure communication path for use by the application functions.*

The highest layer of the Internet protocol stack is the **Application** *layer. This is the home of the protocols that provide such familiar services as Web browsing, e-mail, and file transfer. This is also the layer[3] at which utility services (such as DNP$_3$, IEEE 2030.5, OpenADR, and many others) run.*

3 The Internet model presented here doesn't formally number the layers. The most common alternative, the *Open Systems Interconnection* (OSI) model, defines seven layers, but these layers do not align precisely with those of the Internet model. Speaking loosely, the Link layer corresponds to OSI layers 1 and 2, the Internet layer to OSI layer 3, the Transport layer to OSI layer 4, and the Application layer spans OSI layers five through seven. For this reason, application protocols are sometimes referred to as "Layer 7" protocols.

Rather than avoiding TCP entirely, the WebSocket protocol[4] uses it to support full-duplex communications over a persistent connection. By keeping the TCP connection open, two-way message exchanges can be initiated at any time by either the server or client. Thus, servers can push data to clients without the security risks associated with publicly exposing client addresses. Furthermore, by avoiding the need to create a new connection session for every message exchange, network throughput may be increased, facilitating realtime data transfers such as for grid telemetry. Although WebSocket is not HTTP, it is compatible with the infrastructure that supports HTTP (ports, proxies, and other intermediaries). As such, it is available in most current browsers, including Apple Safari, Google Chrome, Microsoft Edge and Mozilla Firefox.

Reducing Application Protocol Message Sizes

One concern with today's integrated grid application protocols is that they are verbose. Many of them, such as IEEE 2030.5 and OpenADR, label the data in a message using *Extensible Markup Language* (XML).[5] Such systems are used because they greatly simplify interpretation of the text of a message and because they are far less error-prone that positional or comma-separated values in data streams. And an extensible markup language allows new labels to be added to the data stream without breaking existing implementations (which simply ignore any unrecognized labels and associated values). Unfortunately, however, an XML-tagged message often contains many more characters of XML than it does of actual data.

Using OpenADR as an example, a basic message to send 24 hourly prices for a single day contains several thousand characters, almost all of which are XML tags, not the actual data values. In addition to the bandwidth required to send these voluminous labels, it has been found that some commercial OpenADR client devices only have enough memory to store a couple of such messages.

JavaScript Object Notation (JSON) is an alternative to XML that is simpler to use and much more compact, yet still preserves the idea of labeling or tagging each data element for easy interpretation. It's adoption for OpenADR has been suggested; furthermore, a new price communication application being developed for a California utility will offering JSON-formatted messages as a lighter-weight way for client systems to receive price schedules and updates.

Application-Layer "Helpers"

Many functional programs make use of additional application-layer software to facilitate communications. These "helper" technologies, often called "middleware" because it mediates between the functional applications and the lower layers of the network, may provide services like message addressing and routing, error handling, or resource discovery. One common example, HTTP (Hypertext Transport Protocol), is widely used with client-server network architectures. With HTTP, a client system (such as a Web browser) sends a request message to a server (such as a Web site), which responds with both the appropriate functional information as well as various status (completion or error) codes.[6] The use of HTTP allows application writers to use simple commands to send and receive information, without being concerned with the details of establishing connections between systems and processing communication errors. HTTP is one of the most important protocols on the Internet today (it is the basis of the World Wide Web) and it is also widely used by other applications.

The choice of helper software can have a significant impact on the processing and communications burdens placed on edge devices. HTTP is a stateless protocol, meaning that the server does not retain information about clients between requests. However, it is usually run over TCP, and TCP always creates sessions between systems when exchanging messages. Therefore, the TCP layer must create a new session for every HTTP message exchange, which is wasteful of power and bandwidth and introduces communication delays.

4 Internet Engineering Task Force, *The WebSocket Protocol*, RFC 6455 (https://tools.ietf.org/html/rfc6455).

5 A markup language provides a way to "mark up" or label text so that its meaning can be properly determined. It consists of a set of labels or "tags" that are embedded in the data stream to identify the various data values found therein. For example, one such language, HTML (*Hypertext Markup Language*), is used by the Web to tell browsers where to place text, what color and size it should be, and so on.

6 The mode of interaction in which the client system initiates the information exchange by sending a request to the server is called "pull" mode. If the client system is willing to expose its public IP address, then the server can initiate data exchanges (this is called "push" mode). Due to security concerns, HTTP is mostly used in "pull" mode.

Alternative Information Exchange Mechanisms

Some alternatives to HTTP include XMPP (Extensible Messaging and Presence Protocol), MQTT (Message Queuing Telemetry Transport), and CoAP (Constrained Application Protocol). The remainder of this paper will describe these alternatives, which work well with the constrained environments encountered in lower-capability IoT devices that are appearing at the edge of the integrated grid.

Extensible Messaging and Presence Protocol (XMPP)

While HTTP provides a very common method of transporting XML-tagged messages, XMPP[7] provides a lighter-weight alternative. XMPP is an open standard for "massage-oriented middleware," a type of helper application that creates a distributed communications layer to hide the details of the different operating systems and networks on which it is implemented. In addition to native support for transporting XML content, XMPP also supports features, such as service discovery, that may be important to a rapidly-changing IoT environment. XMPP uses a client-server model operating over long-lived TCP connections, thereby avoiding the overhead of continually creating new sessions for each message exchange. XMPP is widely used by such messaging applications as WhatsApp Messenger and Google Talk and is the basis for in-game private chat on PlayStation.

For utility applications, the value of XMPP as an alternative to HTTP has been recognized, for example, by OpenADR, which may be transported over either HTTP or XMPP. Although most OpenADR deployment HTTP, OpenADR servers (called Virtual Top Nodes – "VTNs") are required to support both protocols to be certified as conforming to the specification. A recent European study[8] used OpenADR to implement a Virtual Power Plant (VPP) based on battery charging stations in Germany controlled from a server in Slovenia. In this instance, it was found that the HTTP "pull" mode required excessive bandwidth (due to the client requests for information), while HTTP raised security concerns due to the need to expose the client's public IP address information. XMPP in "pull" mode also suffered from increased bandwidth and latency, so the eventual implementation used XMPP in PUSH mode.

More information on the use of XMPP for IoT applications may be found on the not-for-profit XMPP Standards Foundation.[9]

Message Queuing Telemetry Transport (MQTT)

MQTT is an ISO/IEC[10] and an OASIS[11] standard for "publish-subscribe" messaging middleware. Like XMPP (and HTTP), it operates over TCP and is designed for use with limited storage (via a small code footprint) and limited network bandwidth. As with XMPP, MQTT requires the use of a server for exchanging messages, though in the case of a publish-subscribe model, this server is called a "broker." Messages in MQTT can be as small as two bytes (or as large as 256 MB). MQTT is used by Amazon Web Services IoT offering and for the Microsoft Azure cloud computing IoT Hub service.

All client systems communicate with the broker, which maintains a list of "topics" to which client systems may subscribe. An application that wishes to "publish" some information sends it to the broker and indicates to which topic it is relevant. The broker then sends the information only to those clients that have subscribed to that topic. In this way, publishers do not need to keep track of the clients and their interests: the broker handles those details.

When deploying OpenADR in 2016, Austin Energy grew concerned about the ability of HTTP to scale to anticipated IoT levels, particularly in terms of network bandwidth. They therefore ported OpenADR to run over MQTT and compared its bandwidth consumption with HTTP (both running in pull mode with 10-second polling). Testing over a 30-day period, they found a dramatic bandwidth reduction of nearly an order of magnitude (from 1.29 GB with HTTP to 152 MB with MQTT).

7 XMPP is maintained by the XMPP Standards Foundation (https://xmpp.org/).

8 Kolenc, M., et al., "Virtual Power Plant Architecture Using OpenADR 2.0b for Dynamic Charging of Automated Guide Vehicles," *Electrical Power and Energy Systems*, **104** (2019), pp. 370-382.

9 https://wiki.xmpp.org/web/Tech_pages/IoT_systems

10 ISO/IEC PRF 20922 (https://www.iso.org/standard/69466.html).

11 http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

Constrained Application Protocol (CoAP)

An Internet standard,[12] CoAP was designed to for use on low-power devices connected to unreliable networks of the sort that might be encountered at the edges of the integrated grid. It uses protocols similar to those in the general Internet and allows communications between nodes on the CoAP network and nodes on the general Internet. It can also use the Internet to bridge between CoAP networks. Intended for use with wireless sensor networks, CoAP sends simple binary messages over UDP. The fixed format of the four-byte-long CoAP message header greatly simplifies information extraction.

Although CoAP uses a simple request/response message exchange pattern, it reverses the usual assignment of client and server: the CoAP server instance is installed on the end node, while the CoAP client resides on the *controller* node, which typically manages several end nodes. A mapping has been defined between CoAP and HTTP, allowing standardized access to CoAP resources via HTTP.

What About Security?

Many application protocols rely on the underlying network layers to provide secure connections that guarantee that data is kept private and uncorrupted. When using TCP, a standard security mechanism is Transport Layer Security (TLS),[13] which both authenticates the two end nodes (via public-key cryptography) and encrypts the data flowing over the TCP connections. This mechanism can be used with any of the technologies described above, provided that they use TCP connections.

UDP, which doesn't even guarantee correctness or completeness of messages, also makes no security guarantees. A mechanism called Datagram Transport Layer Security (DTLS), analogous to TLS, has been defined for use with UDP. It is implemented in some Web browsers (such as Chrome and Firefox) to support direct peer-to-peer streaming of audio and video content inside web pages.

XMPP uses TLS-encrypted communications and performs authentication via the IETF Simple Authentication and Security Layer (SASL).[14] MQTT specifies no authentication mechanisms like SASL, and as with many other protocols, relies on the security applied to the underlying transport layer. By default, CoAP runs over the non-secure UDP protocol, but can optionally use DTLS.

Summary and Next Steps

The expanding edges of the integrated grid are reaching ever further into the realms of small, resource-constrained devices and lower-speed, less-reliable networks. Expanding interest in incorporating edge devices into the energy IoT will continue to challenge traditional solutions, and the industry is likely to see increasing adoption of the lighter-weight communications solutions described in this paper. Utilities interested in this so-called "Smart Energy" realm should understand the emerging technologies that will be used to construct the future energy Internet of Things.

As these technologies evolve or are adapted from non-energy domains, an ongoing research and education activity is underway. EPRI is following initiatives in this area (such as OpenFMB [15]) and has an active Technology Innovation project on IoT architectures. Two reports from that project will be published this year. Also, EPRI is considering creating a reference guide to IoT protocols, analogous to the *Protocol Reference Guide*[16] that is currently published and covers DER- and DR-related application protocols.

Acknowledgments

The Electric Power Research Institute (EPRI) prepared this report:

Principal Investigator
Walt Johnson

This report describes research sponsored by EPRI.

12 Internet Engineering Task Force, The Constrained Application Protocol (CoAP), RFC 7252 (https://tools.ietf.org/html/rfc7252).

13 Internet Engineering Task Force, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446 (https://tools.ietf.org/html/rfc8446).

14 Internet Engineering Task Force, Simple Authentication and Security Layer (SASL), RFC 4422 (https://tools.ietf.org/html/rfc4422).

15 More information on OpenFMB can be found at https://openfmb.github.io.

16 *Protocol Reference Guide: Overview of Application Layer Protocols for DER and DR*. EPRI, Palo Alto, CA: 2017. 3002009850. This is the latest edition: an update is currently in preparation for publishing.

Technical Contact

Walt Johnson, Technical Executive
650.855.2013, hwjohnson@epri.com

The Electric Power Research Institute, Inc. (EPRI, www.epri.com) conducts research and development relating to the generation, delivery and use of electricity for the benefit of the public. An independent, nonprofit organization, EPRI brings together its scientists and engineers as well as experts from academia and industry to help address challenges in electricity, including reliability, efficiency, affordability, health, safety and the environment. EPRI also provides technology, policy and economic analyses to drive long-range research and development planning, and supports research in emerging technologies. EPRI members represent 90% of the electric utility revenue in the United States with international participation in 35 countries. EPRI's principal offices and laboratories are located in Palo Alto, Calif.; Charlotte, N.C.; Knoxville, Tenn.; and Lenox, Mass.

Together...Shaping the Future of Electricity

3002013478 December 2018

Electric Power Research Institute
3420 Hillview Avenue, Palo Alto, California 94304-1338 • PO Box 10412, Palo Alto, California 94303-0813 USA • 800.313.3774
• 650.855.2121 • askepri@epri.com • www.epri.com