expresslogic

Industrial Grade X-Ware IoT Platform™

Complete Cloud protocols on top of a secure IPv6 Network Stack for Even the Smallest End Node Devices

Introduction

We've all seen the commercial about the husband and wife lying on the beach at the start of their vacation. She asks him, "You remembered to lock the car, right?" He grabs his phone, opens an App, and uses it to lock the doors of the car and set its alarm, all remotely from the beach.

"Yes, of course," he replies.

What happened is that he accessed the Internet from an App on his phone, and it communicated with his vehicle manufacturer's servers, which then sent a command to his car via the cellular network, locking the doors and activating the alarm. All this was possible through the Internet of Things, or "IoT." For this to be possible, though, several hardware and software technologies had to be brought into play. In this paper, we'll explore some of these technologies, as they relate to enabling "things," like our smartphones, home security devices, thermostats, and medical equipment to interact with us directly or through servers of various types, known as "the Cloud." Note that substantial excerpts from Wikipedia and other sources have been used to help describe these protocols. Where applicable, the sources for such material have been noted.

The Internet of Things

The IoT is a collection of intelligent devices that can intercommunicate with each other and/or with the Internet, and with the Cloud. The idea of the IoT has been around for many years, but until recently, the "things" were limited to computers (mainframes, servers, desktops, and laptops). Its expansion to incorporate small devices is fairly new, but industry analysts project a growth to tens of billions of "things" by 2020. That's an enormous number of devices, and each one of them is a source or consumer of tons of data.

For instance, just in the US alone, in 2015, electric utilities had about 64.7 million smart meters installed, primarily in residential settings. Those meters measure and report on electricity used 24/7, providing daily, even hourly reporting. And that's just the US, and just smart meters. Think of home thermostats, surveillance cameras, traffic signals, and all the other "things" that gather data and send it to the Cloud. This "Big Data" holds valuable information that utility companies and others can analyze to learn more about their customers' usage and thereby better serve their needs. To make this analysis more convenient, a number of analytics providers are offering their own data warehouses, which can store, backup, and secure all this data. In this way, utilities and other Cloud users can avoid the expense of their own Cloud hardware, software, and upkeep, as well as employing analysis tools of their own, or those available

through the Cloud provider. Examples include Amazon Redshift (https://aws.amazon.com/redshift/), and Google BigQuery (https://cloud.google.com/bigquery/).

The Internet was not designed to accommodate this number of nodes, having only a 32-bit field for device address. Fortunately, back in 1994, the Internet Engineering Task Force (IETF) anticipated the need for more IP addresses for the Internet, and initiated the development of a suite of protocols and standards now known as Internet Protocol Version 6 ("IPv6"). IPv6 uses a 128-bit address size compared with the 32-bit system used in IPv4, and allows for as many as $3.4x10^{38}$ possible addresses.[1]

But the expansion of the IoT, and the new small devices that are fueling its expansion, require more than just addressability. Many of these devices have small memory, inexpensive CPUs, are battery powered, yet require real-time responsiveness. Today, we have the hardware technology to create these devices, and small-footprint real-time operating systems (RTOSes) to manage their internal system functions, but common TCP/IP network software protocols are hard pressed to support them, and a new assortment of better suited protocols is needed. For example, TCP might require more memory and/or CPU power than the new ultra-low-cost MCUs that power many small devices can provide.

To address these and other limitations, a number of new connectivity protocols have been developed over the last couple of years. These protocols are intended for use on resource constrained, low-end microcontroller-driven, battery-powered devices found in the home, at work, at play, in industry, and in medical applications. The general goal is to enable these "things" to communicate with each other, with the Internet, and with the Cloud. In some cases, multiple protocols are available for similar functionality, and the choice of the best one is very dependent on the application. The good news is that many protocols are available from many software and system vendors. The bad news is that not all of the implementations are Industrial Grade, ready for mass market use and reliable enough to stand up to the everyday use they might receive in today's highly-automated society. In this paper, we will discuss these protocols, and their availability as part of an Industrial Grade solution.

The most popular of these new connectivity protocols for small devices are **MQTT**, **CoAP, LWM2M, 6LoWPAN, and Thread**. Here is some information on each of these, and a comparison of their differences and relative merits, gathered from the organizations that have defined and implemented them (see footnotes citing sources for this information):

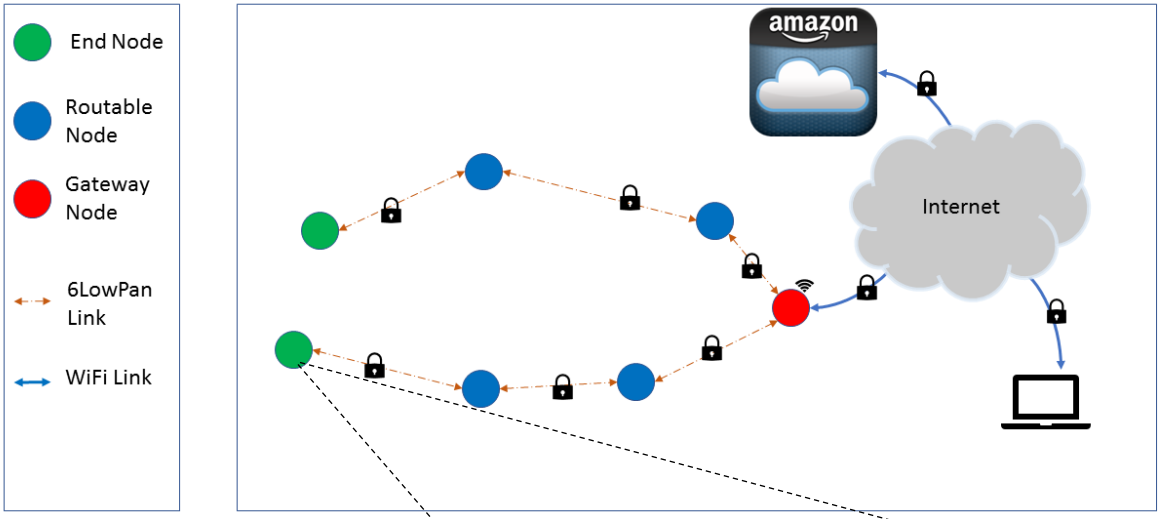
MQTT (MQ Telemetry Transport)

MQTT is a publish/subscribe messaging protocol designed for lightweight M2M communications. It was originally developed by IBM and is now an open standard.

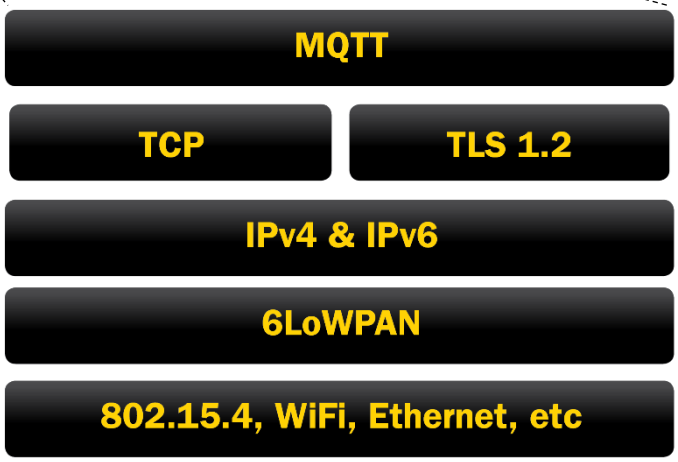
It is designed for connections with remote locations where a small code footprint is required, or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker, responsible for distributing messages to interested clients based on the topic of a message. [2]

[1] https://en.wikipedia.org/wiki/IPv6

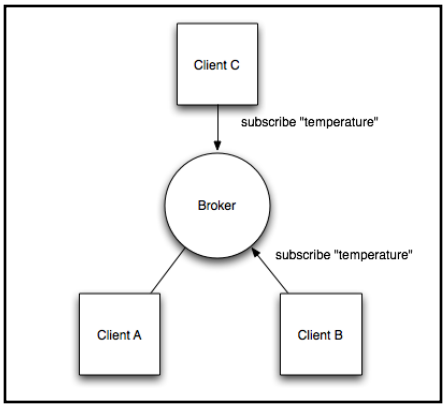
[2] https://en.wikipedia.org/wiki/MQTT

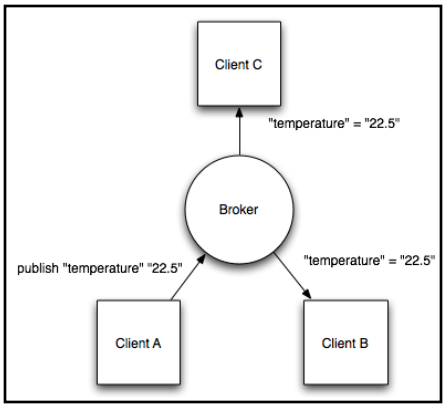


MQTT employs a client/server model, where every sensor end node is a client and connects to a server, known as a broker, over TCP through routable nodes and/or a gateway. The broker typically might be a Cloud service provided by a vehicle manufacturer, as in our example above, or a general-purpose supplier such as Amazon.



For example, imagine a simple network with three clients (A, B, and C) and a central broker. All three clients open TCP connections with the broker. Clients B and C subscribe to the topic “temperature.”





At a later time, Client A publishes a value of 22.5 for topic temperature. The broker forwards the message to all subscribed clients (B and C).

The publisher-subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one. Even though MQTT is designed to be lightweight, it has two drawbacks for very constrained devices.

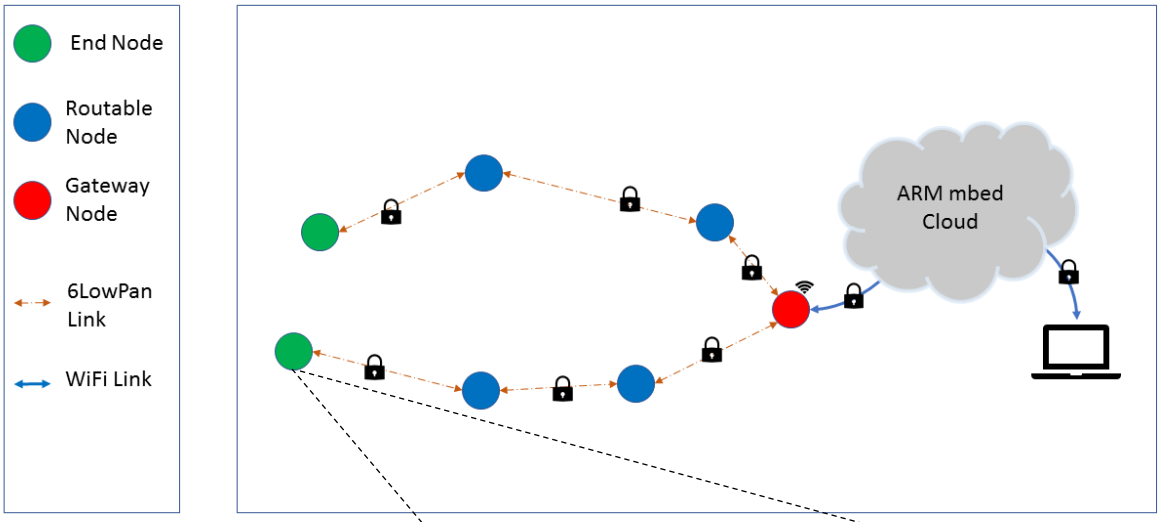
1. Every MQTT client must support TCP and will typically hold a connection open to the broker at all times. For some environments where packet loss is high or computing resources are scarce, this is a problem.
2. MQTT topic names are often long strings which make them impractical for 802.15.4 (low power, low speed wireless radio).

Constrained Application Protocol (CoAP)

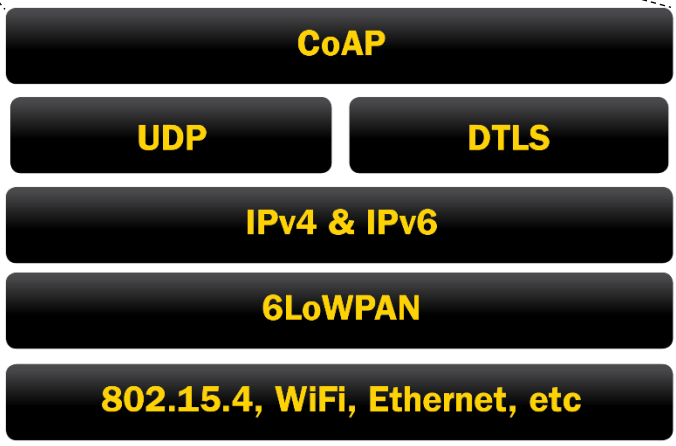
CoAP is an Internet Application Protocol for constrained devices. It enables those constrained devices to communicate with the Internet using similar protocols. CoAP is designed for use between devices on the same constrained network, between devices and general nodes on the Internet, and between devices on different constrained networks, joined by an internet.

CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity. Unlike MQTT, which requires TCP, CoAP uses the smaller and simpler UDP, which is extremely important for resource constrained IoT devices.[3]

[3] https://en.wikipedia.org/wiki/Constrained_Application_Protocol



Express Logic has implemented these IoT protocols (CoAP, UDP, IPv6, and 6LoWPAN) in just 25KB of code. This configuration supports a device end-node, with limited IPv6 and 6LoWPAN functionality. The 25KB code size does not include the ThreadX RTOS (6KB) nor the optional DTLS (5KB), or any application code. It represents only the code necessary for cloud communication.



MQTT/CoAP Comparison

Both MQTT and CoAP protocols are very useful, but they each have pros and cons. Choosing the right one depends on your application:

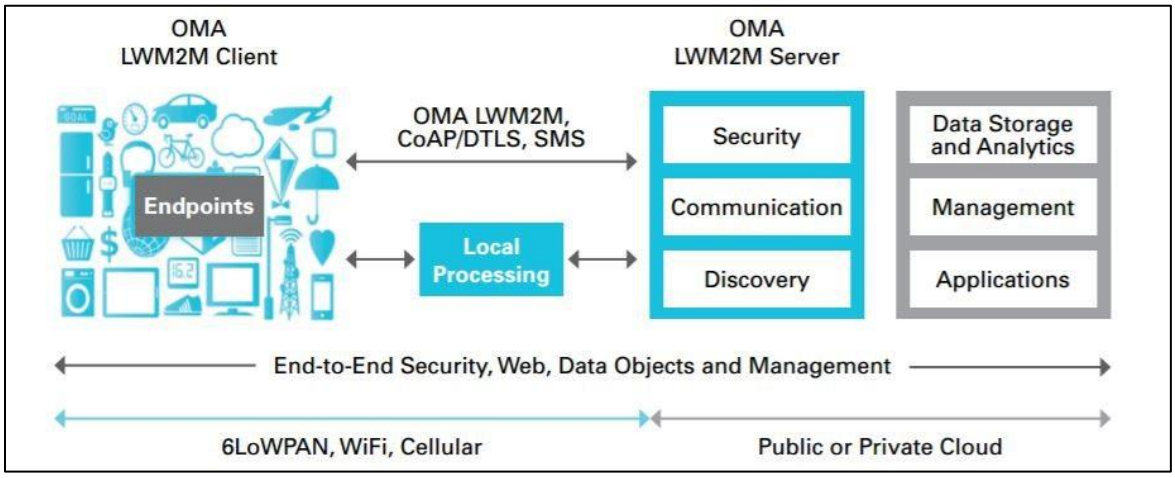
- **MQTT:**
  - Requires TCP, which is larger than UDP, making MQTT not as small as CoAP
  - MQTT offers publish/subscribe semantics (on the same socket) which makes it easier to program on the IoT device side. IoT cloud service providers like AWS IoT and Everything and others offer MQTT based device connectivity.
  - It requires a message broker (server) for its functioning. This makes it a good option for remote/cloud communication, since the cloud server acts as the message broker between the IoT device and other app/services.
  - This also makes it NOT a great option for local network communication between devices, because it requires the end-user to deploy an additional broker in the system.

- **CoAP:**
  - CoAP runs on UDP and thus can be run on extremely resource constrained environments.
  - It is a good mechanism for local network communication, particularly when there is an ecosystem of other CoAP devices.[4]

[4] https://www.quora.com/Which-is-the-best-protocol-to-use-for-IOT-implementation-MQTT-CoAP-XMPP-SOAP-UPnP

Lightweight M2M (LWM2M)

LwM2M is an open industry protocol from the Open Mobile Alliance (“OMA”), built to provide a means to remotely perform service enablement and application management for IoT embedded devices and connected appliances. It is a communication protocol for use between client software on an M2M device and server software on a M2M management and service platform.



source: OMA

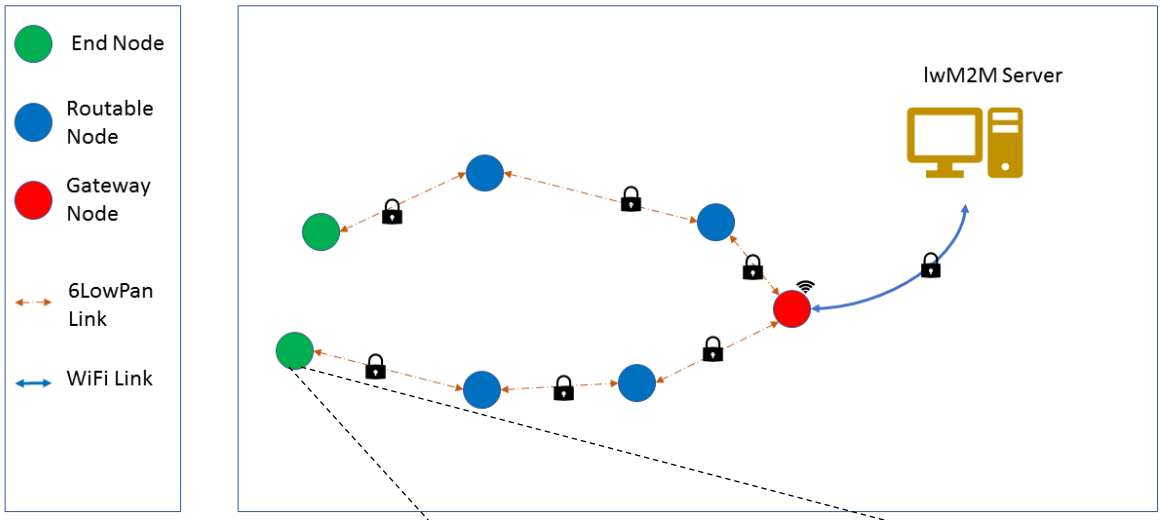
The standard was created as a response to the industry’s desire for a low-cost remote management and service enablement mechanism that works over wireless connections and is lightweight, according to OMA. The group says the motivation to create LWM2M was the need to overcome issues from technical fragmentation, find a suitable mechanism to cater to the needs of constrained M2M devices, and to generate benefits from decoupling system components via standardized interfaces.

The LWM2M protocol has four main characteristics:

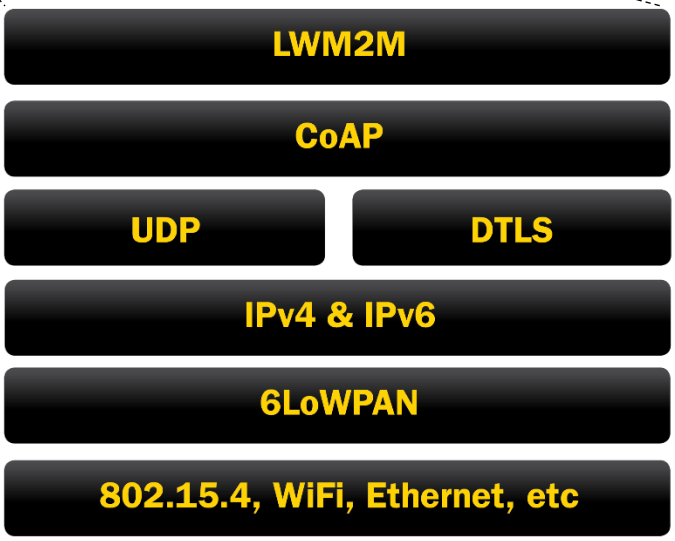
1. An architectural design based on a representational state transfer application protocol interface;
2. It defines a resource and data model;
3. It has been designed with performance and the constraints of M2M devices in mind;
4. It reuses and builds on the constrained application protocol secure data transfer standard that has been standardized by the IETF as a variation of the internet’s HTTP protocol (appropriate for data transfer to and from low-cost connected IoT devices). [5, 6]

[5] http://www.rcrwireless.com/20161212/internet-of-things/lwm2m-tag31-tag99

[6] http://www.openmobilealliance.org/wp/Overviews/lightweightm2m_overview.html



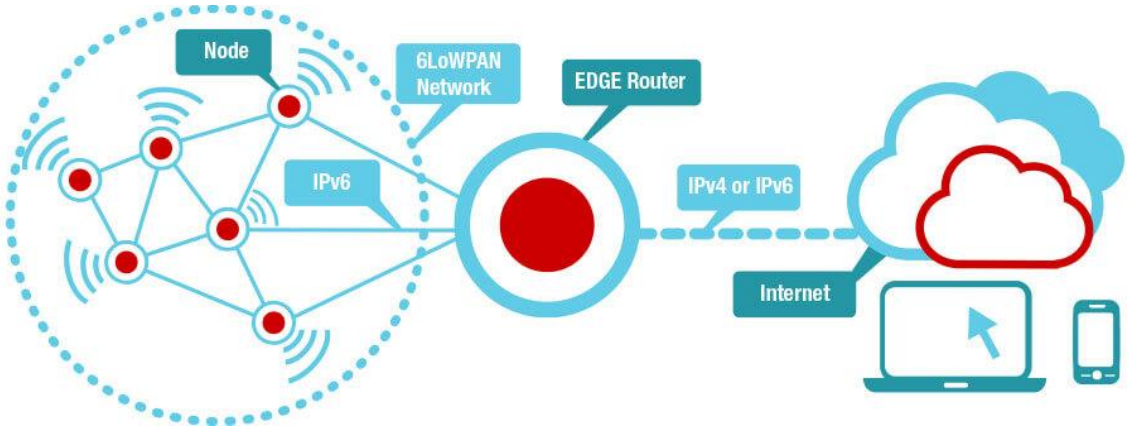
With LWM2M, the end-node application's interface to Amazon AWS, for example, is through the CoAP protocol. For messages sent to Amazon AWS, the message will go from LwM2M to CoAP, through UDP, TLS/DTLS, and IPv6. If ethernet is the physical transport path for the device, the IPv6 packet goes out directly via the Ethernet driver. If 802.15.4 is the physical transport path of the device, the message is sent out over 6LoWPAN and the 802.15.4 driver.



6LoWPAN

6LoWPAN is an acronym of *IPv6 over Low Power Wireless Personal Area Networks*. The 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices," and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.

6LoWPAN is a mesh network protocol, allowing IPv6 datagrams to be transmitted by low-power short-range radio (such as IEEE 802.15.4). A low-power radio typically has a span of 30 feet, with low bandwidth (in the kilobits per second range), and small packet size (128 bytes). 6LoWPAN bridges IPv6 and the low-power radio network.



6LoWPAN provides:

- Open standards including TCP, UDP, HTTP, COAP, MQTT, and websockets
- End-to-end IPv6 addressable nodes
- No need for a gateway or proxy. A 6LoWPAN border router connects the 6LoWPAN network to the Internet
- One-to-many and many-to-one routing
- Robustness and scalability
- Use across multiple communications platforms (i.e. Ethernet / Wi-Fi / 802.15.4 / Sub-1GHz ISM)
- Interoperability at the IP level[7]

[7] http://www.ti.com/lsds/ti/wireless-connectivity/6lowpan/overview.page

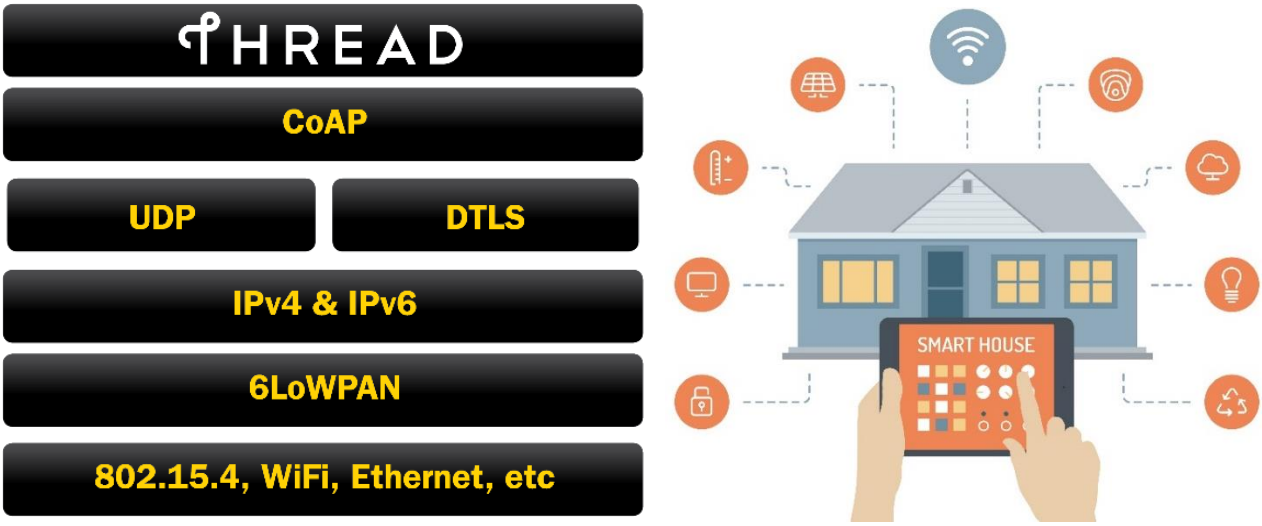
Thread

Thread is an IPv6-based, royalty-free networking protocol for smart home automation devices to communicate on a local wireless network.

The "Thread Group" alliance was formed in 2014, and today is a working group including Nest Labs (a subsidiary of Alphabet/Google), Samsung, ARM, Qualcomm, NXP, Silicon Labs, Somfy, OSRAM, Tyco International, and the lock company Yale, in an attempt to have Thread become the industry standard by providing Thread certification for products.

Thread uses 6LoWPAN, which in turn uses the IEEE 802.15.4 wireless protocol with mesh communication, as does ZigBee and other systems. Thread however is IP-addressable, with Cloud access and AES encryption. It currently supports up to 250 devices in one local network mesh.[8]

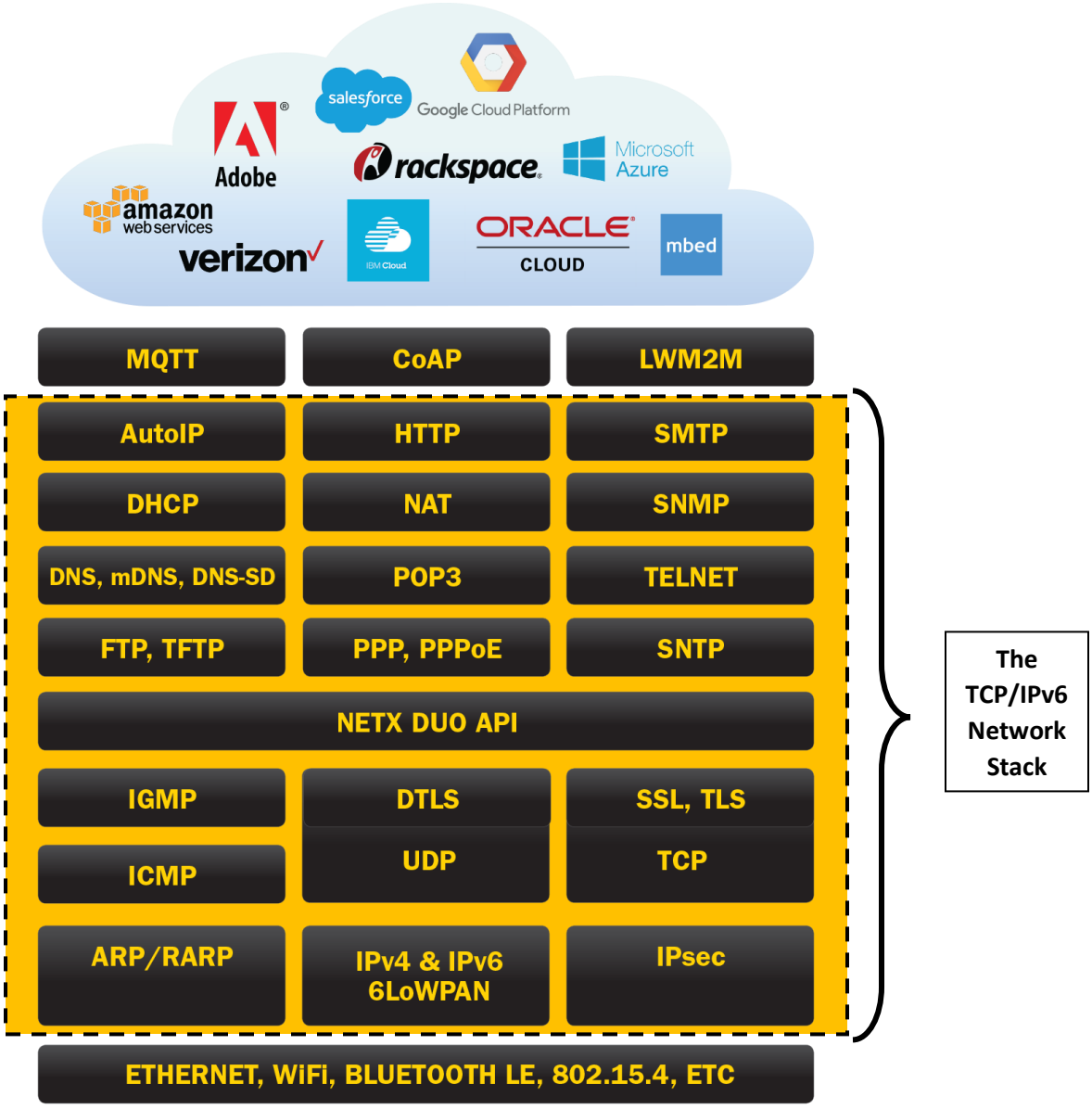
The Connected Home



[8] https://en.wikipedia.org/wiki/Thread_(network_protocol)

The Critical Underlying TCP/IPv6 Network Stack

Each of the protocols described above relies on an underlying IP Stack for IPv6 communication. The new protocols are designed at the application level (except for 6LoWPAN), without regard for the means of transport (ie; Ethernet, WiFi, or Cellular). As such, much of the “heavy lifting” is relegated to the IP Stack, which as it turns out is significantly larger than the Cloud protocols themselves.



It’s not surprising, then, that the underlying IP Stack is much more complex, and much more critical to the exchange of information, even though it is more general in design, and not specifically tailored for the Cloud. When choosing Cloud protocols, it is equally, if not more

critical to select the right IP Stack on which these Cloud protocols will rely for proper and efficient operation.

Industrial Grade

Of course, the IP Stack must support the IPv6 protocol. Ideally, this support should be validated and certified by some independent authority. Beyond that, the Cloud protocols should be tightly integrated with the IP Stack, to assure efficiency and correctness of operation under all demanding use cases. The IP Stack must be "Industrial Grade," ready for production use, not designed just to meet the needs of beginners, hobbyists, or academic projects. It should be SMALL, SAFE, SECURE, ADVANCED, FAST, and EASY-TO-USE:

- **Small**: It would be of no ultimate benefit for a Cloud protocol to be small in size so it could fit within the memory constraints of a low-cost microcontroller, if the underlying IP Stack were too big itself. The IP Stack must be small as well, so as not to interfere with the goal of the small Cloud protocol.

- **Safe**: The IP Stack should satisfy popular safety standards for electronic device software, including IEC 61508 SIL 4, IEC 62304 Class C, ISO 26262 ASIL D, UL/IEC 60730, UL/IEC 60335, UL 1998, and EN 50128 SW-SIL 4. This assures its ability to be certified for use in safety critical systems, as well as being beneficial for use in other systems.

- **Secure**: The IP Stack should be closed – with external access defined by the application, not the stack itself. It should also support security protocols such as IPsec, TLS, SSL, and DTLS.

- **Advanced**: The IP Stack should offer advanced technology like the ability to communicate with IPv4 as well as IPv6, hardware checksum support where available, optional application protocols beyond the Cloud, such as AutoIP, DHCP, DNS, and mDNS.

- **Fast**: Performance and efficiency of the IP Stack is critical to its mission. It must be able to operate at "near wire-speed," the theoretical maximum of the transport hardware, lest it introduce overhead that interferes with its mission.

- **Easy-to-Use**: It must be designed from the ground up for ease of use, with an intuitive API, clean, clear source code, and help developers get products to market faster than less capable stacks.

The Industrial Grade X-Ware IoT Platform™

Express Logic's X-Ware IoT Platform™ is built on NetX Duo, just such an IP Stack. NetX Duo also supports IPv4 at the same time, enabling it to serve networks of mixed mode. It meets all the above criteria for an Industrial Grade solution, and has been field proven to deliver all the expected benefits.

In addition to the dual-stack IPv4/IPv6 NetX Duo, X-Ware Platform for the IoT includes Express Logic's popular, widely deployed ThreadX RTOS, found in over 5.4 billion embedded systems, and all the IoT Protocols mentioned above (the Thread protocol will be available in Q4 of 2017).

X-Ware IoT Platform is Industrial Grade, 100% designed, integrated, and supported by Express Logic, and supports many MCU platforms, including those based on the ARM Cortex-M0+. Its small size enables "Device to Cloud in Under 25KB."

Following is a summary of the characteristics of the NetX Duo IP Stack:

NetX Duo is Small

Today's IoT nodes employ low-cost MCUs with limited RAM and Flash. Large stacks are not able to fit into such devices, making a small-size almost a must-have for the IoT.

- Minimal Flash (IP/UDP) – 14KB
- Minimal RAM – 3.3KB
- Automatic Scaling, includes only those service functions referenced by the application
- Able to execute from Flash, no need to copy to RAM
- Coded to achieve small component sizes, for example:
    - TCP – 12.3KB
    - ICMP – 2.5KB
    - UDP – 2.1KB

NetX Duo is Fast

High performance is the result of low overhead, efficient design and coding. Low cost MCUs do not have abundant processing power, making efficiency and low overhead all that more important. Performance of frequently used services can become a significant burden, if using a stack that is not coded to achieve maximum efficiency.

- Efficiently coded to achieve nearly the theoretical maximum transfer rate for a communications medium (eg: 10K/100K/1G ethernet) "Near Wire Speed" @ 120MHz:
    - Iperf UDP Receive: 95Mbps
    - Iperf UDP Transmit: 94Mbps
    - Iperf TCP Receive: 92Mbps
    - Iperf TCP Transmit: 93Mbps

NetX Duo is Advanced

Advanced technology enables developers to address challenges in a more efficient manner than writing new code for such requirements, or licensing multiple products to cover all needs:

- Dual IPv4/IPv6 Support in the same stack. Use either, or both, from the same application
- Zero Copy Implementation, avoiding overhead for most packet transfers
- Hardware Checksum Offload Support, speeding up computation, and freeing the CPU
- Hardware Crypto Support, where available, for higher performance
- Multihomed Support
- Static Routing Support
- Extensive Add-on Protocol Support:
  AutoIP, DHCP, DNS, mDNS, FTP, HTTP, IPsec, NAT, POP3, PPP, SMTP, SNMP, Telnet, TFTP, TLS/DTLS
- Unlimited Network Objects

NetX Duo is Safe

NetX Duo is one of the few TCP/IP network stacks to have been certified for use in safety-related systems by the IEC and UL. To be certified, NetX Duo had to pass rigorous review and validation testing, and had to demonstrate analytically correct coding:

- Pre-Certified to Many Safety Standards:
  IEC 61508 SIL 4, IEC 62304 Class C, ISO 26262 ASIL D, UL/IEC 60730, UL/IEC 60335, UL 1998, EN 50128 SW-SIL 4
- IXIA IxANVL Validated (RFC Compliant)
- Phase-II IPv6 Ready Logo Certified
- Full Source Code
  - Coverity Clean
  - C-STAT Clean
- Vast Processor/Tool Support

NetX Duo is Secure

In today's world, data and device security is of paramount concern. NetX Duo has been designed to achieve high-security inherently, and through the availability of several protocols to serve a variety of security needs, all of which are developed and fully supported by Express Logic:

- Closed System – External Access Defined by Application
- IPsec
- TLS
- DTLS
- SSL

NetX Duo is Easy-To-Use

Ease of use is a strong factor in achieving fast Time To Market. NetX Duo employs the same intuitive API design as the rest of Express Logic's X-Ware, making it easy to learn and correctly use:

- Intuitive API
- Great Documentation – User Guide Written First!
- Automatic Configuration
- Optional BSD Socket API
- High-Quality Source Code

Summary

The IoT is exciting, both for consumers and for vendors of technology that enables the design and development of the kinds of products that consumers want. The new Cloud protocols extends the IoT from Device to Cloud, and are best suited for use with small-memory, limited-performance Microcontrollers. To do so, the Cloud protocols must be implemented in a small, efficient fashion, and importantly, must be designed for use on to of a capable, small, fast IP Stack. Express Logic's X-Ware IoT Platform is just such a Device to Cloud solution. Not only does it incorporate all of the new protocols for communication with the Cloud, but it is built on top of the Industrial Grade X-Ware Platform™, with its NetX Duo IPV4/IPv6 dual TCP/IP stack. Be sure to make a careful evaluation of the underlying IP Stack before committing to any Cloud-only solution. With X-Ware IoT Platform, it's all pre-integrated and complete: Device to Cloud in Under 25KB!