IoT Devices and Network Connectivity

CHRISTIAN LEGARE | JULY 17 2017
Christian E. Legare

Director, IoT OS Platform
Silicon Labs

Chairman and President
IPSO Alliance
Enabling the IoT at the Device Level

A collaborative group of organizations committed to enabling the IoT at the device level by outlining the semantic, protocol, identity, services, physical, autonomous and resource models to define a smart object and ensure smart object interoperability

www.ipso-alliance.org
Connected Device System versus IoT
IoT Building Blocks

IoT = Identification + Sensing + Communication + Computation + Services + Semantics

<table>
<thead>
<tr>
<th>Application Protocol</th>
<th>DDS</th>
<th>CoAP</th>
<th>AMQP</th>
<th>MQTT</th>
<th>MQTT-SN</th>
<th>XMPP</th>
<th>HTTP/REST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Discovery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DNS-SD</td>
</tr>
<tr>
<td>Routing Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RPL</td>
</tr>
<tr>
<td>Network Layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IPv4 / IPv6</td>
</tr>
<tr>
<td>Link Layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IEEE 802.15.4</td>
</tr>
<tr>
<td>Physical / Device Layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z-Wave</td>
</tr>
<tr>
<td>Infrastructure Protocols</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical / Device Layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z-Wave</td>
</tr>
<tr>
<td>Influential Protocols</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEEE 1888.3, IPSec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IEEE 1905.1</td>
<td></td>
</tr>
</tbody>
</table>
**IoT Systems – Consumer vs Industrial**

Even though this industry is very young, we are starting to see the emergence of two types of IoT systems

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Industrial IoT</th>
<th>Human IoT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Market Opportunity</strong></td>
<td>Brownfield (known environment)</td>
<td>Greenfield (uncharted domain)</td>
</tr>
<tr>
<td><strong>Product Lifecycle</strong></td>
<td>Until dead or obsolete</td>
<td>Whims of style and/or budget</td>
</tr>
<tr>
<td><strong>Solution Integration</strong></td>
<td>Heterogeneous APIs</td>
<td>Vertically integrated</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>Access</td>
<td>Identity &amp; privacy</td>
</tr>
<tr>
<td><strong>Interaction</strong></td>
<td>Autonomous</td>
<td>Reactive</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>0.9999 to 0.99999 (4–5 ‘9’s)</td>
<td>0.99 to 0.999 (2–3 ‘9’s)</td>
</tr>
<tr>
<td><strong>Access to Internet</strong></td>
<td>Intermittent to independent</td>
<td>Persistent to interrupted</td>
</tr>
<tr>
<td><strong>Response to Failure</strong></td>
<td>Resilient, fail-in-place</td>
<td>Retry, replace</td>
</tr>
<tr>
<td><strong>Network Topology</strong></td>
<td>Federations of peer-to-peer</td>
<td>Constellations of peripherals</td>
</tr>
<tr>
<td><strong>Physical Connectivity</strong></td>
<td>Legacy &amp; purpose-built</td>
<td>Evolving broadband &amp; wireless</td>
</tr>
</tbody>
</table>

Source: Patrick Morehead, Forbes, “Who Wins In The Industrial Internet Of Things (IIoT)?”, October 29 2013
Relative size of IoT components in number of units

- Backend Systems (e.g., Cloud)
- Gateways (Fog) Edge routers
- Things
Relative size of IoT components in number of units

- Backend Systems
  i.e. Cloud

- Gateways
  Edge routers

- Things
Relative size of IoT components in number of units

- **Backend Systems** i.e. Cloud
- **Gateways** Edge routers
- **Things**
Relative size of IoT components in number of units

- **Backend Systems**
  - i.e. Cloud
- **Gateways**
  - Edge routers
- **Things**
“We sometimes underestimate the influence of little things...”
Charles W. Chesnutt

The Springtail is smaller than the head of a pin. There are 200,000 per square meter of soil.
Ant-sized IoT Radio Created by Stanford University

Includes antenna, synchronization, computation. Implemented in 65nm CMOS. Uses power-harvesting. Transmits 60 GHz pulses, range up to 50 cm.
Local Networking Technologies
Local Networking Technologies

In a Wireless Sensor Network (WSN) node, the networking technology is usually a short-range wireless access link. There are other IoT end devices that do not use WSN but other types of networking technology.

**Wired**
- Ethernet, EtherCAT, EtherIP
- Modbus, Profinet
- DASH7
- HART
- HomePlug, GreenPhy, G.hnn (HomeGrid)
- And more…

**Wireless**
- Bluetooth
- Wi-Fi
- Zigbee, Zigbee IP and other 802.15.4 capable protocols such as 6LoWPAN
- ANT, Z-Wave
- DASH7
- ISA100: Wireless Systems for Industrial Automation, Process Control and Related Applications
- Wireless HART
- EnOcean
- Wireless M-Bus
- And more…
Wireless LAN

- WLAN Selection Criteria
- Protocol Efficiency
- Power Efficiency
  - How long will my battery last?
- Peak Power Consumption
- Performance
- Range
- Robustness
- Throughput
- Latency
- Coexistence
Memory Requirements
A Few Short Range Wireless Examples

6LoWPAN stacks range in code sizes of 50-100 kB typically. RAM size depends on the buffer approach used, but it is typically a few kB of RAM.

Coordinator/Router - 116 kB flash 7 kB RAM
End Device - 99 kB flash 3.8 kB RAM
## TCP/IP

### ROM

<table>
<thead>
<tr>
<th>Options</th>
<th>IPv4 Only</th>
<th>IPv6 Only</th>
<th>IPv4 &amp; IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Enabled</td>
<td>66.5 kB</td>
<td>75.4 kB</td>
<td>93.8 kB</td>
</tr>
<tr>
<td>TCP Disabled</td>
<td>44.4 kB</td>
<td>53.2 kB</td>
<td>70.0 kB</td>
</tr>
</tbody>
</table>

### RAM

<table>
<thead>
<tr>
<th>Configuration</th>
<th>IPv4 Only</th>
<th>IPv6 Only</th>
<th>IPv4 &amp; IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>16.7 kB</td>
<td>17.8 kB</td>
<td>18.5 kB</td>
</tr>
<tr>
<td>Typical</td>
<td>42.0 kB</td>
<td>44.2 kB</td>
<td>45.5 kB</td>
</tr>
</tbody>
</table>

- Fully RFC compliant
- Interrupt driven
There are multiple ways to use Wi-Fi:

- Wi-Fi station
- Wi-Fi Access Point (AP, needs a DHCP server)
- Wi-Fi Direct (WD, needs a DHCP server)
- Wi-Fi Protected Setup (WPS)

Wi-Fi stack typical footprint is about 200 KB for AP, STA and WD.
802.15.4, Wi-Fi, Bluetooth and others

Running the wireless network on-board or off-board?

- On-board requires the manufacturer to FCC certify the product
- Using an off-board radio module is more expensive but is certified
- A similar approach is applicable to many wired or wireless interfaces

**WiFi example**

**Solution 1 – Low volume**
- Run the TCP/IP stack and Wi-Fi stack off board

**Solution 2 – Med volume**
- Run the TCP/IP stack on the host target and the Wi-Fi supplicant on the Wi-Fi module

**Solution 3 – High volume**
- Run the TCP/IP stack and Wi-Fi stack on the target board
Bluetooth Classic

300 to 400 KB is an average footprint (influenced by the number of profiles included, and required feature set)
Bluetooth Smart
Around 200 KB is an average footprint
Dependent on feature set

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Fitness</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Alert Proximity</td>
</tr>
<tr>
<td>Auxiliary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generic Access Profile</th>
<th>GATT (Generic Attribute Profile)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Manager Protocol</td>
<td>Attribute Protocol</td>
</tr>
</tbody>
</table>

L2CAP
Host Controller Interface

<table>
<thead>
<tr>
<th>Link Manager</th>
<th>Direct Test Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Layer</td>
<td></td>
</tr>
</tbody>
</table>
Cellular WAN

Cellular modem for the WAN

TCP/IP stack requires PPP to connect to the cellular modem (small code and RAM add-on)

Modem needs to be approved by the cellular carrier

Some possible modems:
- Telit
- Gemalto
- Sirerra Wireless
- ...

Cellular modem for the WAN

TCP/IP stack requires PPP to connect to the cellular modem (small code and RAM add-on)

Modem needs to be approved by the cellular carrier

Some possible modems:
- Telit
- Gemalto
- Sirerra Wireless
- …
Wireless LAN Coexistence

**Bluetooth LE and Wi-Fi spectrum usage**

Each channel is 2MHz wide with no wasted spectrum

**Zigbee and Wi-Fi spectrum usage**

Each channel is 2MHz wide with a wasteful 5MHz spacing. In the presence of Wi-Fi, only 4 channels are likely to be available.

Source: CSR application note CS-213199-AN
The long term forecast puts Bluetooth LE and WiFi as the predominant WLAN technologies.

Expect 6LoWPAN (Thread) to run on Bluetooth radios.

**Legend**
- LE (Bluetooth low energy)
- A (ANT)
- A+ (ANT+)
- RF (RF4CE)
- Zi (ZigBee)
- Wi (Wi-Fi)
- Ni (Nike+)
- Ir (IrDA)
- NF (NFC)

Source: CSR application note CS-213199-AN
Java Usage in IoT

- There are around 450,000 embedded software engineers using C/C++
- There are 9 million Java developers in the world
- ARM, Oracle and Freescale believe Java is the solution to the creation of an IoT ecosystem

Java ME Memory footprint (approximate)

From: 350KB ROM, 130 KB RAM (for a minimal, customized configuration)
To: 2000 KB ROM, 700 KB RAM (for the full, standard configuration)

- It is too large for the typical IoT device processor
- Java virtual machines for smaller processor exist, but not from Oracle
The Thing
To plan for the software and hardware requirements, the Thing design must take into consideration the environment:

- The variables to measure or control (analog interface)
- The frequency at which these variables processing needs to be done (processor performance)
- The transport technology of these variables value and variable control (wired or wireless)
- Sensing, processing and networking software memory requirement
- The availability for a secure device firmware upgrade (SDFU)
  - Secure bootloader RAM and Flash requirement
  - SDFU can be remote [over the air] (networking capabilities required, Firmware Over the Air : FOTA)
- The availability for device management (Commissioning [ID and Auth], Provisioning, Fault and Service management)
Power vs Performance

- 8/16-bit processors are still very popular for sensor devices because of their low power consumption
- 32 bit devices are required to integrate IP networking capabilities
- MCUs with integrated radio are becoming mainstream
- ARM is designing a compatible Cortex-M0 with close to threshold voltage of CMOS transistors, and at clock frequencies of the order of tens of kilohertz.
Typical Thing Processor

The Average MCU

- 50 to 200 Mhz CPU Clock
- 64KB to 2MB Flash (code space)
- 4KB to 512KB RAM
IoT Device / WSN Node
Two-Processor Solution

Sensor/Actuator

Processor

HW - 8/16 bit processor
SW - Foreground/Background Application

HW - 32 bit processor
SW - Real-Time Kernel
LAN or WAN

Processor

Communication
IoT Device / Sensor Node
One-Processor Solution

Sensor/Actuator
Hardware: 32 bit processor

Processor

Communication
Software: Real-Time Kernel Application LAN or WAN
Consumer IoT Device

Application
- Real-Time Kernel
- Power Management
- Remote Device Firmware Upgrade

Java Virtual Machine

Vertical Market-Specific Protocol

Sensors

Bluetooth

TCP/IP Ethernet/Wi-Fi

Bluetooth Radio / Wi-Fi Radio

TCP/IP Ethernet/Wi-Fi

3G/4G radio

Ethernet

Actuators

Typical:
- Cortex-M3/M4
- Cortex-A

Optional

One of the Possible Networks
- AllSeen
- HomePlug/HomeGrid
- Continua Alliance
- 2net
- Industrial Internet
A sensor node can be written as a foreground/background (single thread / super loop) system

When networking is involve, the use of a real-time kernel is highly recommended
- Best usage of the processor time
- Better software architecture
- Simplified development
- Simplified maintenance

Code mainly written in C

Java is also an option, but not from Oracle
- The Oracle JVM is too large for the average IOT device (thing)
Low Power (or not) IoT Device / WSN Node
Single Processor Solution

Assumption: 6LoWPAN runs on the BTLE radio
WSN Edge Node or Gateway

Communication
- LAN
- 6LoWPAN
- Bluetooth
- Wi-Fi
- Ethernet
- and more...

Processor
- HW
- 32 bit processor
- SW
- Real-Time Kernel

Communication
- WAN
  - Access to Internet Service Provider:
    - Wi-Fi
    - Ethernet
  - Cellular
With LAN and WAN present in an edge node/gateway, it is mandatory to use of a real-time kernel or other form of OS.
- Best usage of the processor time
- Better software architecture
- Simplified development
- Simplified maintenance

- Code mainly written in C

- Java is also an option
  - With a Cortex-A class processor, Oracle JVM is possible
  - With a Cortex-M class processor, the Oracle JVM is too large. Another solution is required and exist.
Gateway / WSN Edge Node

- Bluetooth Radio / Wi-Fi Radio
- Application
  - Real-Time Kernel
  - Power Management
  - Remote Device Firmware Upgrade
  - Java Virtual Machine
- Vertical Market Specific Protocol
- Ethernet / Wi-Fi / 3G-4G
  - HTTP Client
  - MQTT Client
  - CoAP Client
  - TCP/IP
  - Bluetooth
  - 6LowPAN

**Typical:**
Cortex-M3/M4
Cortex-A

**One of the possible protocols**

- AllSeen
- HomePlug/HomeGrid
- Continua Alliance
- 2net
- Industrial Internet
IoT Protocols
IP Usage

Web Services

IP family of protocols can be used to provide services to device

- Examples: SMS text, e-mail, file sharing, streaming audio, speech to text, social media ...
IP Usage

IoT Services

The availability of back-end services based on IP protocols are what is differentiating “IoT devices” from “connected devices”:

Storage, multiple devices/applications data usage, system analytics and potential for efficiency gain...
# Internet Protocol Types

<table>
<thead>
<tr>
<th>Request/Response</th>
<th>Publish/Subscribe</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>MQTT</td>
</tr>
<tr>
<td>WebSocket</td>
<td>CoAP</td>
</tr>
<tr>
<td>CoAP</td>
<td>XMPP</td>
</tr>
</tbody>
</table>

- **HTTP** and **WebSocket** can be used for **Request/Response** communication. *WebSocket* also supports **polling**.
- **CoAP** is used for **IoT Services**.
- **MQTT** and **XMPP** are used for **Publish/Subscribe** communication.

Can do Request/Response with polling
Request/Response

HTTP
HTTP Request – The Notion of REST

HTTP client opens a connection (socket) and sends a request message to an HTTP server.

The server then returns a response message, usually containing the resource that was requested.
After delivering the response, the client closes the connection (making HTTP a stateless protocol, that is, not maintaining any connection information between transactions).

This is where the notion of REST (REpresentational State Transfer) comes from.
HTTP and HTTPS

• Typical HTTPS (showing the messages, not the number of packets)
• Areas in blue are optional (bidirectional SSL/TLS)
• Connection is initiated by a client
• Client always has to poll the server, server cannot initiate connection: not efficient for an embedded device
• High overhead: Open/Send/Close for every application message
Request/Response

Websocket
WebSockets are a bi-directional, full-duplex, persistent connections from a client to a server.

Once a WebSocket connection is established the connection stays open until the client or server decides to close this connection.
A Websocket Connection

With this open connection, the client or server can send a message at any given time to the other.

This makes web programming entirely event driven, not (just) user initiated.

It is stateful.
Publish/Subscribe

XMPP
XMPP

- Extensible Messaging and Presence Protocol
- Runs over TCP, and sometimes over HTTP over TCP
- Key strength is the use of the name@domain.com addressing scheme that leverages the global DNS infrastructure making it easy to find devices on the Internet
- XMPP mainly use polling
- Servers can push using BOSH (Bidirectional streams over Synchronous HTTP)
Web Versus Dedicated IoT

**Web**
- Hundreds / thousands of bytes
- XML
- HTTP
- TLS
- TCP
- IPv6
  - Inefficient content encoding
  - Huge overhead, difficult parsing
  - Requires full Internet devices

**Internet of Things**
- Tens of bytes
- Web Objects
- CoAP
- DTLS
- UDP
- 6LoWPAN
  - Efficient objects
  - Efficient Web
  - Optimized IP access
Publish/Subscribe

CoAP
CoAP – Design Goals

• Constrained devices
  Processor - Flash/RAM

• Constrained Networks
  i.e., Wireless Sensor Networks

• Low power devices
  (sleep modes)

• Caching/Mapping to HTTP

• Resource processing

• Subscribe/Notify architecture

• Resource discovery

• Multicast

• UDP Transport

• Reliable

• Low Latency

• Use of MIME types

• Manageability
CoAP Architecture

RFC 7252

Proxy/Gateway

Constrained device
As of July 2016, the IETF draft defining publish/subscribe and message queuing functionality for CoAP that extends the capabilities for supporting nodes with long breaks in connectivity and/or up-time is in its 5th iteration.

https://datatracker.ietf.org/doc/draft-koster-core-coap-pubsub/
Publish/Subscribe

MQTT
MQTT

- MQTT has a client/server model, where every device is a client and connects to a server, known as a broker, over TCP.
- MQTT is message oriented. Every message is a discrete chunk of data, opaque to the broker.
- Every message is published to an address, known as a topic. Clients may subscribe to multiple topics.
- Every client subscribed to a topic receives every message published to the topic.
MQTT

MQTT Version 3.1.1 was last revised or approved by the membership of OASIS on 29 October 2014.

- "OASIS" (Organization for the Advancement of Structured Information Standards) is a non-profit consortium that drives the development, convergence and adoption of open standards for the global information society.

- IBM and Microsoft are amongst the initial foundational founders. The list of members can be found at this link: https://www.oasis-open.org/member-roster

- Excellent training source: http://www.hivemq.com/
MQTT Quality of Service (QoS)

There are 3 QoS levels in MQTT:

- **QoS 0**
  - at most once
  - Does not survive network failure
  - Never duplicated

- **QoS 1**
  - at least once
  - Survives network failure
  - Can be duplicated

- **QoS 2**
  - exactly once
  - Survives network failure
  - Never duplicated

The Quality of Service (QoS) level is an agreement between sender and receiver of a message regarding the guarantees of delivering a message.
MQTT is lightweight but has two drawbacks for very constrained devices:

- 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.
- MQTT topic names are often long strings which make them impractical for 802.15.4.

Both of these shortcomings are addressed by the MQTT-SN protocol, which defines a MQTT UDP mapping and adds broker support for indexing topic names.
DDS
DDS is Decentralized

DDS
- Is an Object Management Group (OMG) standard
- Was introduced in 2004
- Uses a Publish/Subscribe architecture
- Uses network resources efficiently
- Commercial and Open Source versions available

Can be deployed without servers/brokers

- Fast
  - 100,000’s update/sec
- Scalable
  - Load independent # apps
- Managed with QoS
- Reliable
  - No single point of failure

DDS Data-Centric Bus

App  App  App  App  App  App  App
CIP
Common Industrial Protocol (CIP)

- CIP Motion™ Profiles
- Motor Control Profiles
- Transducer Profiles
- I/O Profiles
- Other Profiles
- Semiconductor Profiles
- CIP Safety™ Profiles

Object Library
(Communications, Applications, Time Synchronization)

- Safety Object Library

Data Management Services
Explicit and I/O Messages

- Safety Services and Messages

Connection Management, Routing

TCP/IP

- EtherNet/IP™
- CompoNet™
- ControlNet™
- DeviceNet™

Network Applications of CIP
One More Level

Device Management
Device Management

One of the most important requirement for an IoT system

- Very much Cloud based

Device Management

- Configure the device (provisioning)
- Update the firmware securely (and maybe the application)
- Monitor and gather connectivity statistics

Out of the multiple device management functions, security and provisioning are the most complex ones
Device Management

Existing Systems/Protocols

- **TR-069** (well known for broadband modem management, SOAP based)

- **OMA-DM** (An Open Mobile Alliance [OMA] standard for Device Management, mainly used by mobile network operators)

- **Lightweight M2M** (new OMA standard)
LWM2M Architecture

- Built on top of CoAP
- MQTT also used now (Eclipse projects)
- Much lighter than OMA-DM and TRS-069
Benefits of OMA Lightweight M2M

- Simple, efficient protocol, interfaces and payload formats
- Transport security based on DTLS
  - With Pre-shared and Public Key modes, Provisioning and Bootstrapping
- Powerful Object and Resource model
  - Global registry and public lookup of all Objects
  - Provides application semantics that are easy to use and re-use
  - Standard device management Objects already defined by OMA
Benefits of OMA Lightweight M2M

- Applicable to Cellular, 6LoWPAN, Wi-Fi and ZigBee IP or any other IP based constrained devices or networks
- Ideal time-to-market for the standard
  - LWM2M is commercially deployed since 2013
  - Can be combined with existing Device Management offerings
  - Will be supported in OneM2M and can be integrated with ETSI M2M
Messaging Standards

What messaging protocols do you use for your IoT solution?

- HTTP: 61.2%
- MQTT: 52.4%
- CoAP: 21.2%
- HTTP/2: 19.2%
- In-house / proprietary: 15.5%
- AMQP: 13.9%
- XMPP: 13.2%
- I don’t know: 7.4%
- Proprietary vendor protocol: 6.2%
- Other: 5.3%
- DDS: 3.5%
- None: 2.3%
Questions?

Thank you!