Sécurité de l'Internet des Choses

Pascal.Urien@Telecom-ParisTech.fr

About the Internet of Things (IoT)

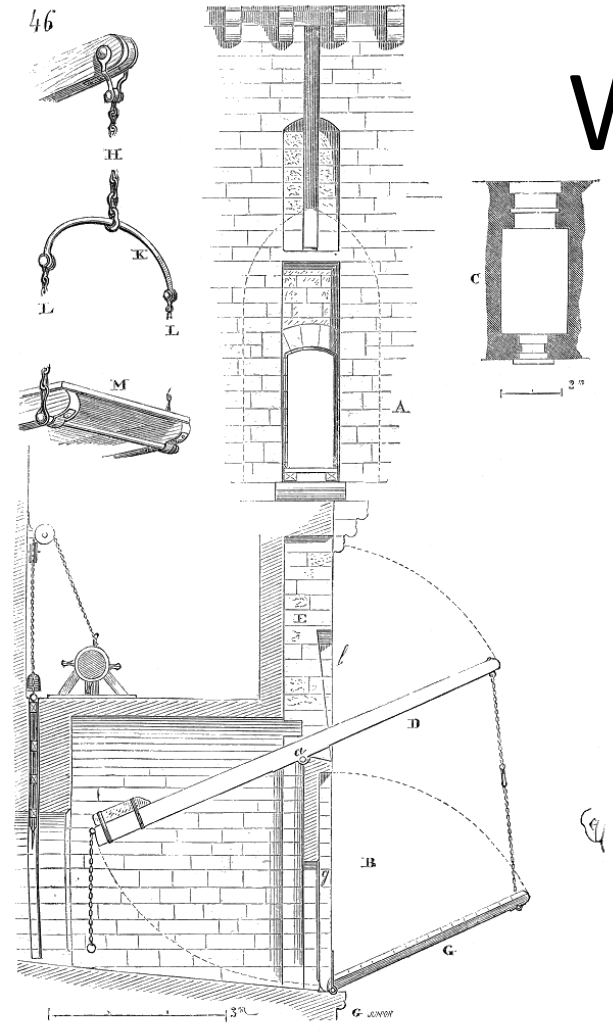
- Pretz, K. (2013). “The Next Evolution of the Internet”

The *Internet of Things* (IoT) is a *network of connected things*.

Objet: Chose solide considérée comme un tout, fabriquée par l'homme et destinée à un certain usage

Appareil: Objet, machine, dispositif électrique, électronique, mécanique, etc., formés d'un assemblage de pièces destinées à fonctionner ensemble

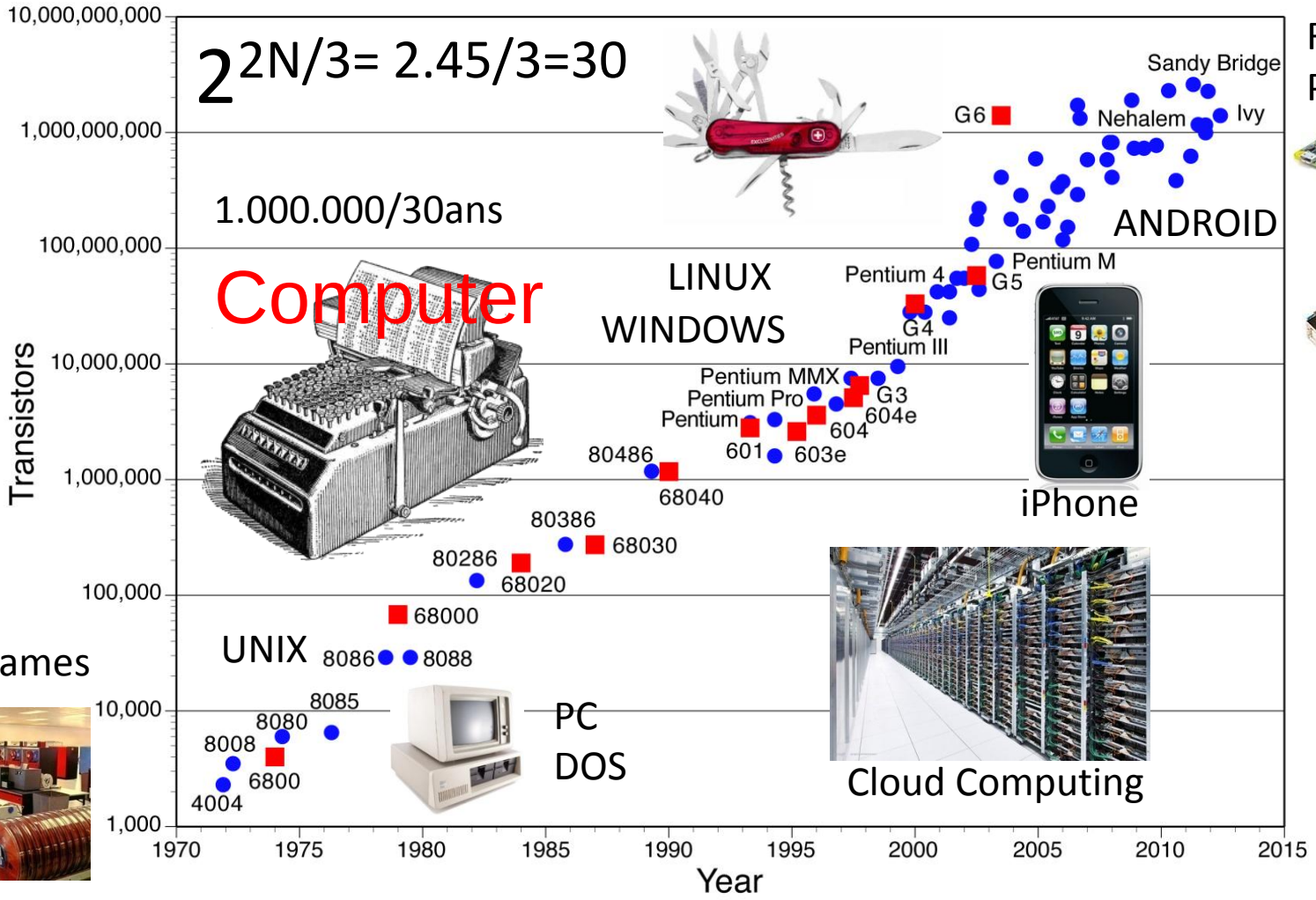
Machine: Appareil ou ensemble d'appareils capable d'effectuer un certain travail ou de remplir une certaine fonction, soit sous la conduite d'un opérateur, soit d'une manière autonome.



What is a Thing?

- A computer
  - CPU
  - Memories (RAM, ROM, EEPROM, FLASH…)
  - IO buses
- With at least one network interface
  - Wi-Fi, Bluetooth, ZigBee…
- Equipped with sensors and actuators

8-bit Atmel Microcontroller
- 64/128/256KB Flash
- 4KB EEPROM
- 8KB SRAM
- Peripheral Features


$2^{2N/3}$= 2.45/3=30
1.000.000/30ans
Computer
LINUX
WINDOWS
UNIX
PC
DOS
ANDROID
iPhone
Cloud Computing
Main Frames
Raspberry Pi
Arduino
Transistors
Year
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
Sandy Bridge
Nehalem
Ivy
G6
Pentium M
Pentium 4
G5
G4
Pentium III
Pentium MMX
Pentium Pro
G3
Pentium
604e
604
601
603e
80486
68040
80386
68030
80286
68020
68000
8086
8088
8085
8080
8008
6800
4004

Beyond The Horizon

- The IoT is the death of the Moore Law.
- Waldrop M. "More Than Moore", Nature February 2016 Vol 530
  - *The semiconductor industry will soon abandon its pursuit of Moore's Law.*

Beyond The Horizon

- “Rebooting the IT Revolution: A Call to Action" (SIA/SRC), 2015
  - "*Security is projected to become an even bigger challenge in the future as the number of interconnected devices increases... In fact, the Internet of Things can be viewed as the largest and most poorly defended cyber attack surface conceived by mankind*“
  - "*a short list of requirements includes tamper resistance and secure communications and storage*".

"*A short list of requirements includes tamper resistance and secure communications and storage*"

Node Integrity

Isolation

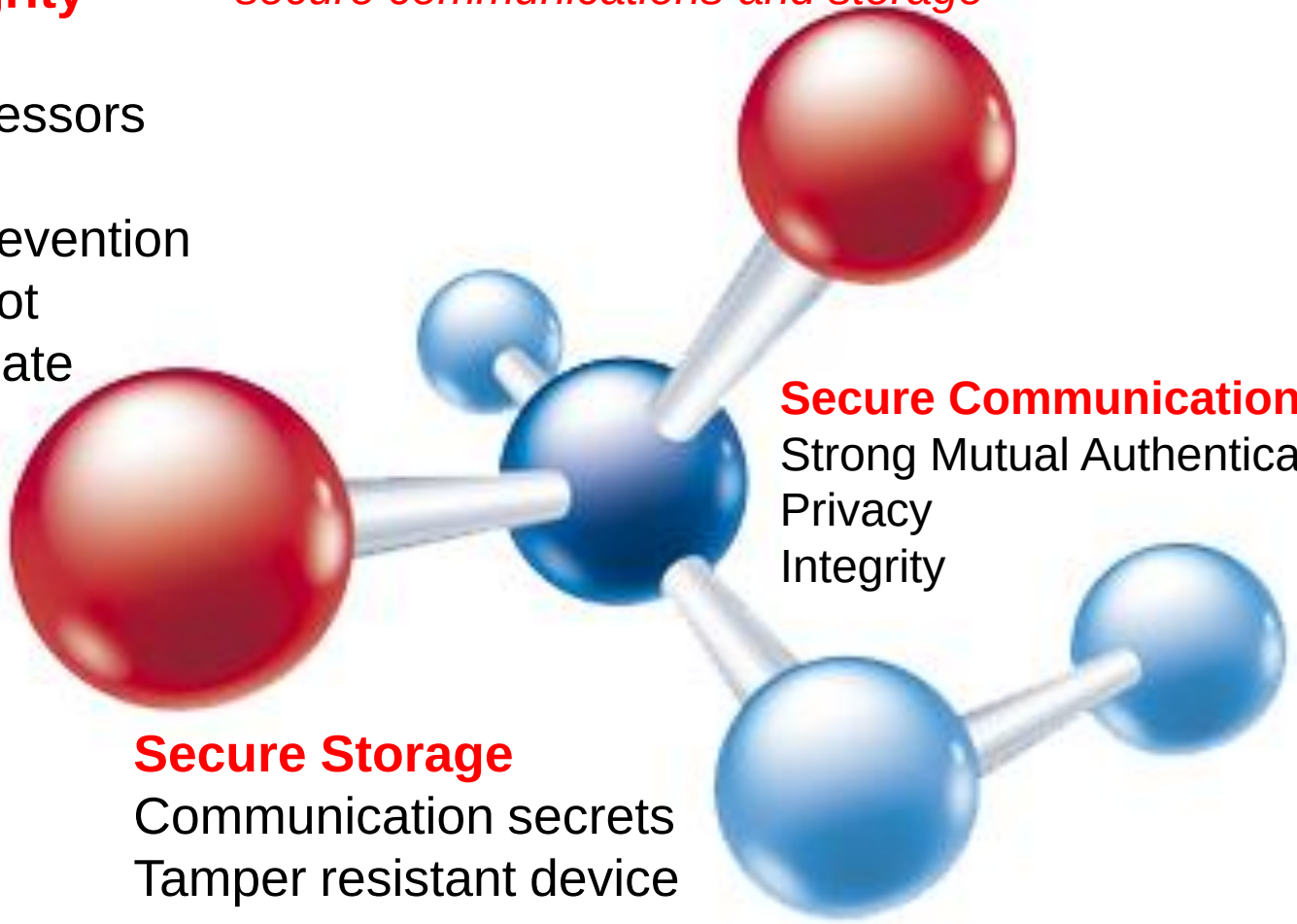
-Multi processors

- Sandbox

Intrusion prevention

-Secure Boot

Secure update



Secure Communication

Strong Mutual Authentication

Privacy

Integrity

Secure Storage

Communication secrets

Tamper resistant device

Secure Communications, 2015

It is time to recap what we have.

There is an undocumented telnet port on the IP camera, which can be accessed by default with root:123456, there is no GUI to change this password, and changing it via console, it only lasts until the next reboot.

I think it is safe to tell this a backdoor.

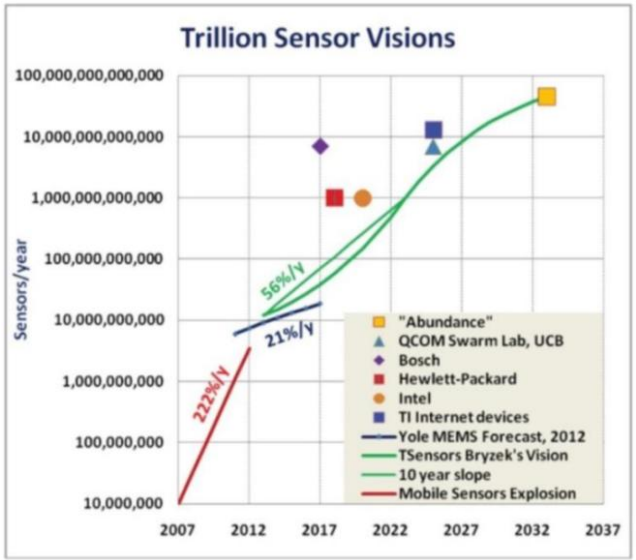
…Last but not least everything is running as root, which is not surprising.

September 2016. Mirai Malware
145.607 cameras
1 terabit/s
35,000/50,000 HTTP request/s
25,000 IP adresses
More than 100 countries

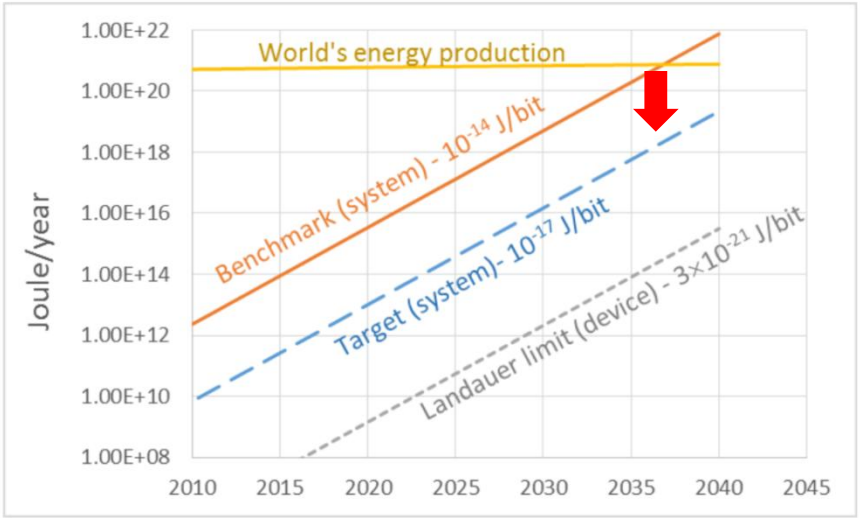
Trillion Sensors

*W= ½ Nq x V
q = 1,6 10^{-19}
10^{-14} J == 125,000 electrons

- In current mainstream systems, the lower-edge system-level energy per one bit *transition is ~10^{-14} J, which is referred as the "benchmark".



Towards Cyber Physical Systems (CPS)

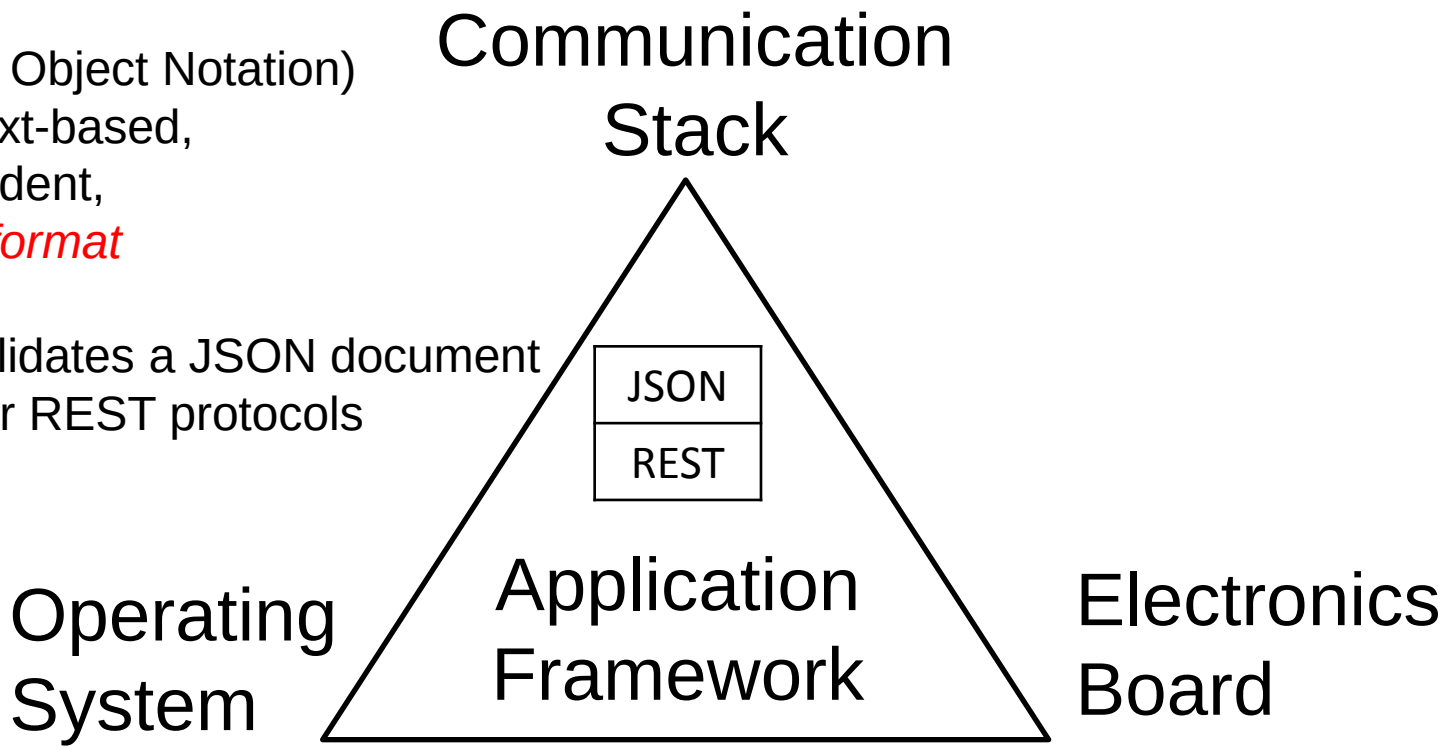


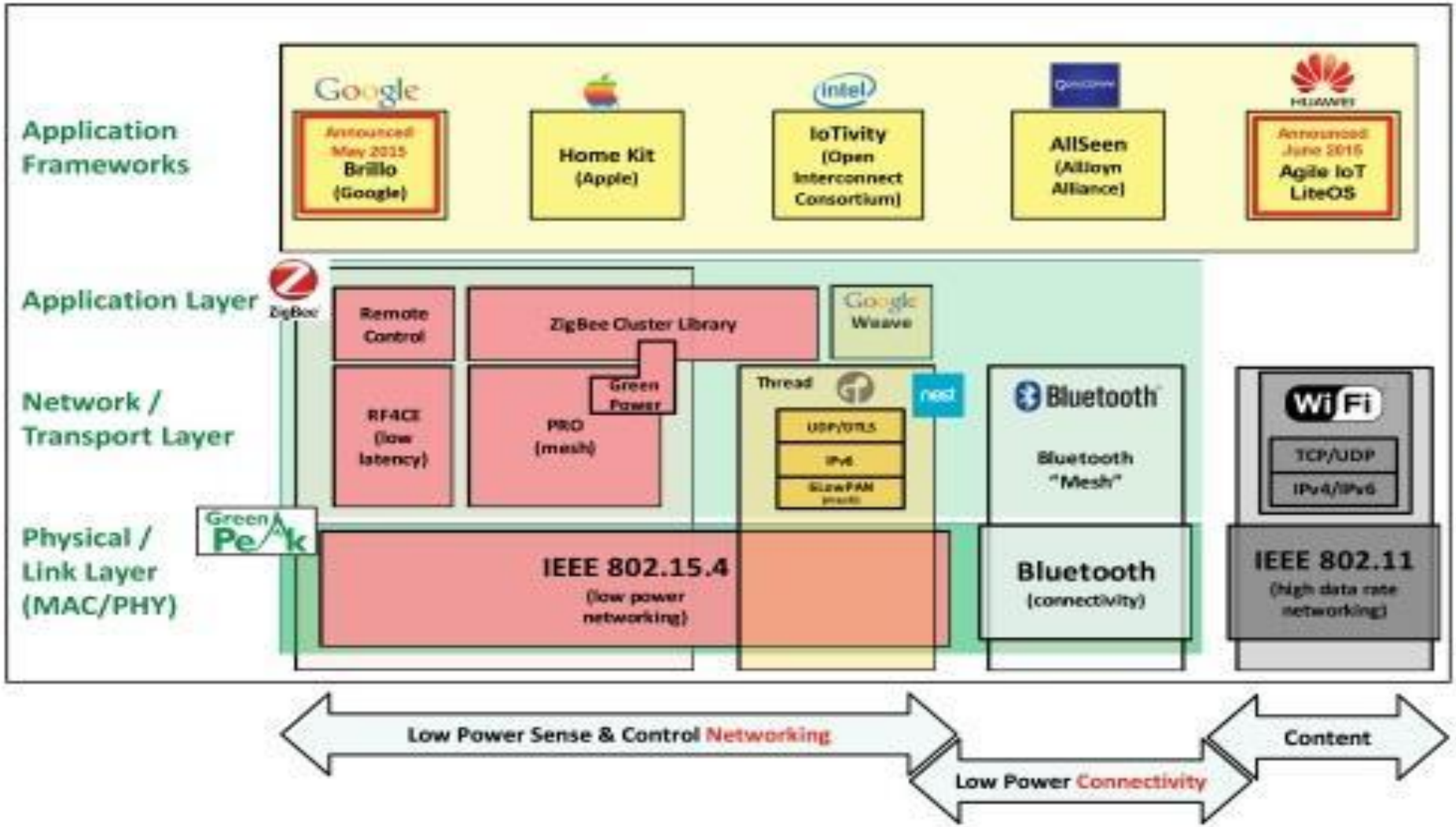
Internet Of Things

JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent, *data interchange format*

JSON Schema validates a JSON document
JSON is used over REST protocols

Linux, Contiki, Riot, Iotivity, AllJoyn, Brillo, mbed OS ...)




Application Frameworks
Google
Announced May 2015
Brillo
(Google)
Home Kit
(Apple)
intel
IoTivity
(Open Interconnect Consortium)
AllSeen
(Alliayn Alliance)
HUAWEI
Announced June 2015
Agile IoT LiteOS
Application Layer
ZigBee
Remote Control
ZigBee Cluster Library
Google Weave
Network / Transport Layer
RF4CE (low latency)
PRO (mesh)
Green Power
Thread
nest
UDP/DTLS
IPv6
6LowPAN
Bluetooth
Bluetooth "Mesh"
WiFi
TCP/UDP
IPv4/IPv6
Physical / Link Layer (MAC/PHY)
GreenPeak
IEEE 802.15.4
(low power networking)
Bluetooth
(connectivity)
IEEE 802.11
(high data rate networking)
Low Power Sense & Control Networking
Low Power Connectivity
Content

IoT Systems

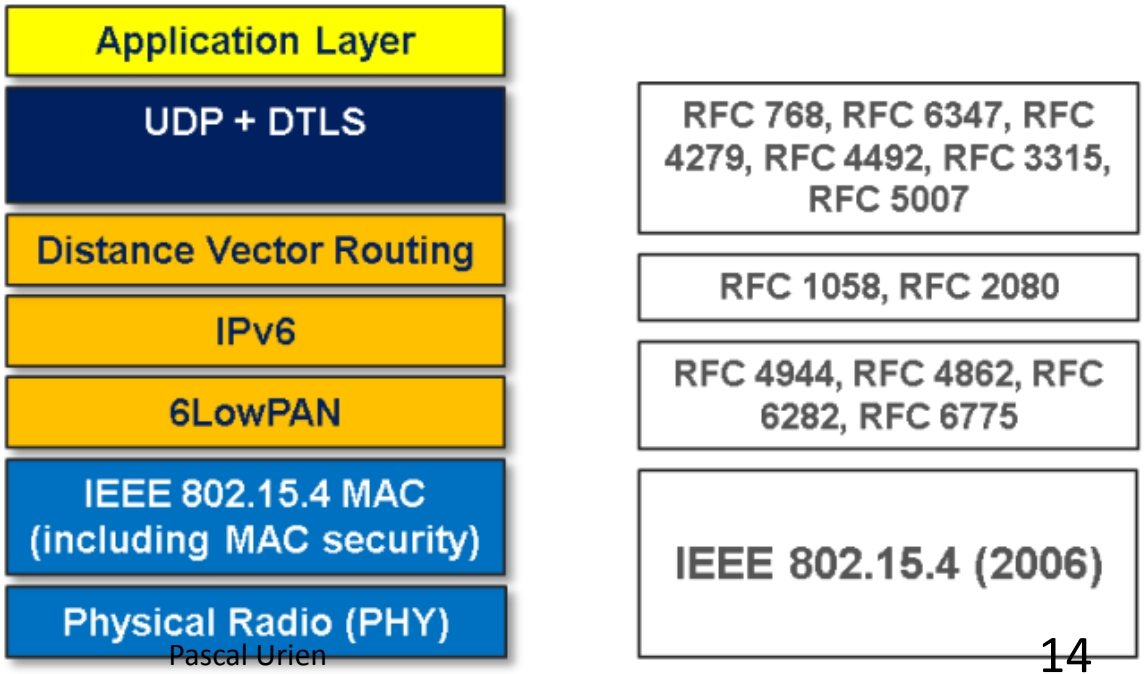
- Thread
  - 6LowPAN, DTLS+Password, Commissionner-Joiner architecture, supported by NEST boards
- Open Connectivity Foundation (OCF)
  - 6LowPAN, DTLS+Authentication, Access Control List (ACL), REST API, Iotivity framework
- MBED stack from the ARM company
  - IPv4, 6LoPAN, TLS/DTLS, HTTP, CoAP, MQTT, LWM2M. IBM KIT
- The HAP (*HomeKit Accessory Protocol*) from Apple
  - BlueTooth, Wi-Fi, HTTP, JSON, application security, Secure Remote Password procedure (SRP, RFC 5054).
- Brillo and Weave from Google
  - Brillo is an OS, 35MB footprint. Weave is a communications platform. 802.15.4 (zigbee, threads), BLE, Wi-Fi, Ethernet. HTTPS. Schema Driven (JSON)associates Weave XMPP requests with application function invocations. OAuth 2.0 Authentication, Google as Authentication Server (AS). Intel® Edison Board.
- Philips Hue Bulbs
  - ZigBee Light Link (ZLL). A same link key is shared by all nodes. Bridge with IP/UDP interface.
- Amazon Dash Button
  - Wi-Fi, Bluetooth, HTTPS, Mobile phone as a bridge with AWS

Example 1. Thread

https://www.threadgroup.org

DTLS + J-PAKE Authentification

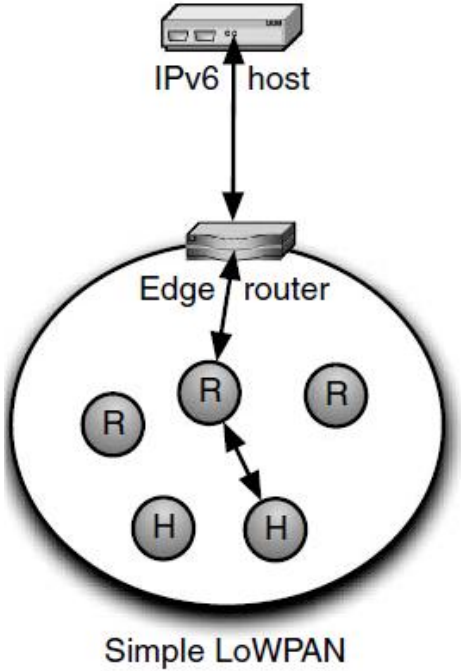
J-PAKE is a password-authenticated key exchange (PAKE) with “juggling” (hence the “J”).
It essentially uses elliptic curve Diffie-Hellmann for key agreement and Schnorr signatures as a NIZK (Non-Interactive Zero-Knowledge) proof mechanism

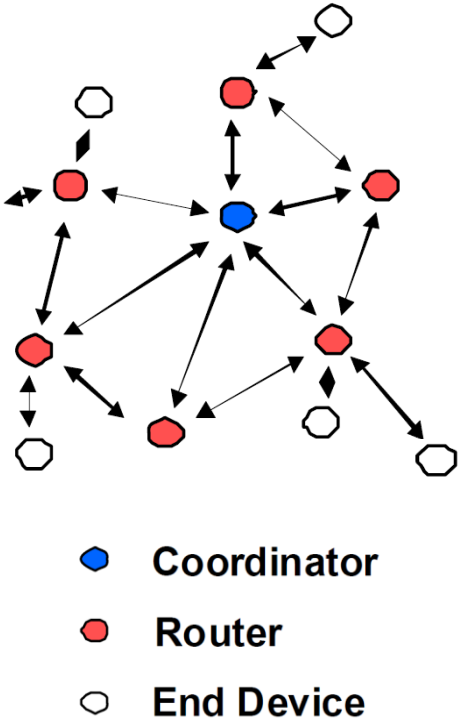


6LoWPAN = IPv6 + Adaptation Layer

IEEE 802.15.4
MAC Frame Size 127 Bytes
IPv6 header 40 Bytes
TCP header 20 Bytes

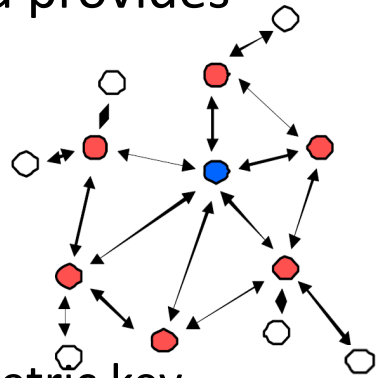
IEEE 802.15.4. Segmentation/Assembly operations are performed by an Adaption Layer and two kinds of routing mechanisms are supported *mesh-under* (performed in the adaptation layer) and *route-over* (performed in the IPv6 layer).





IEEE 802.15.4

- Coordinator is assumed to be the Trust Center (TC) and provides
  - Cryptographic key establishment
  - Key transport
  - Frame protection
  - Device management
- Cryptographic Keys
  - **Master Key** , basis for long term security used for symmetric key establishment. It is used to keep confidential the Link Keys exchange between two nodes in the Key Establishment Procedure (SKKE).
  - **Link Key**, shared between two network peers for Unicast communication.
  - Network Key, used for broadcast communication security.



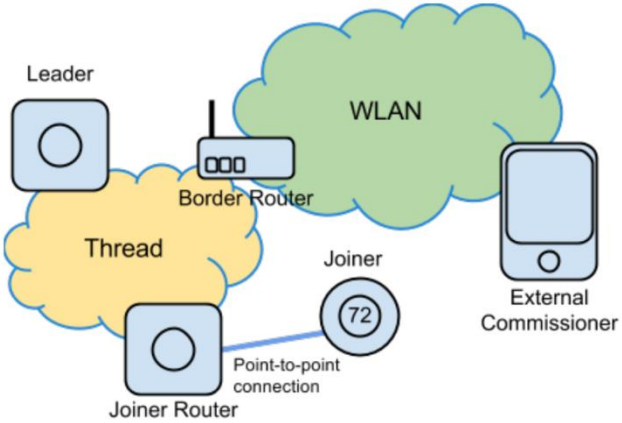
Thread Entities

- Border Router
  - interface point for the Commissioner when the Commissioner is on a non-Thread Network.
- Commissionner
  - The currently elected authentication server for new Thread devices and the authorizer for providing the network credentials they require to join the network.
- Petitioning
  - The process of authenticating and authorizing a Commissioner Candidate onto the Thread Network through a representative (typically the Border Router).

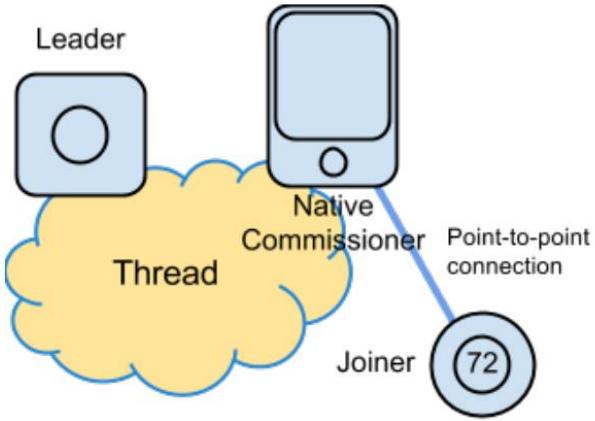
Thread Entities

- Joiner
  - The device to be added by a human administrator to a commissioned Thread Network. The Joiner does not have network credentials.
- Joiner Router
  - An existing Thread router or REED (Router-Eligible End Device) on the secure Thread Network that is one radio hop away from the Joiner.
- KEK
  - Key Establishment Key used to secure delivery of the network-wide key and other network parameters to the Joiner.
- Leader
  - The device responsible for managing router ID assignment.

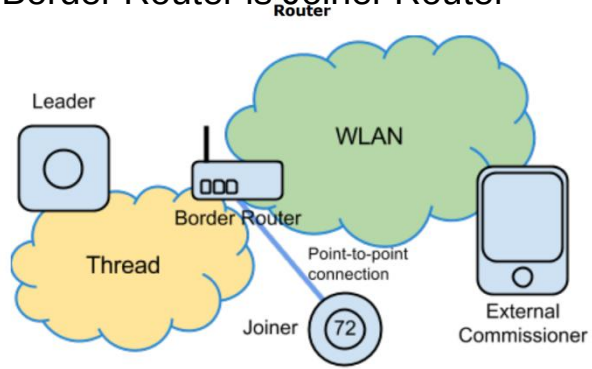
Case 1: External Commissioner connected to the WLAN, Border Router is not Joiner Router



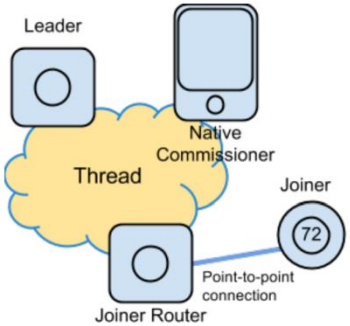
Case 4: Native Commissioner connected to Thread Network, Joiner Router is Commissioner



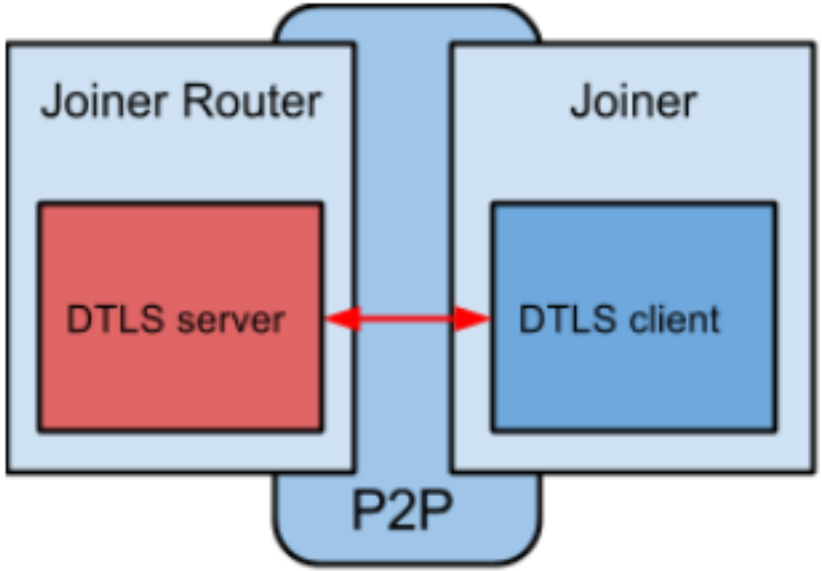
Case 2: External Commissioner connected to the WLAN, Border Router is Joiner Router

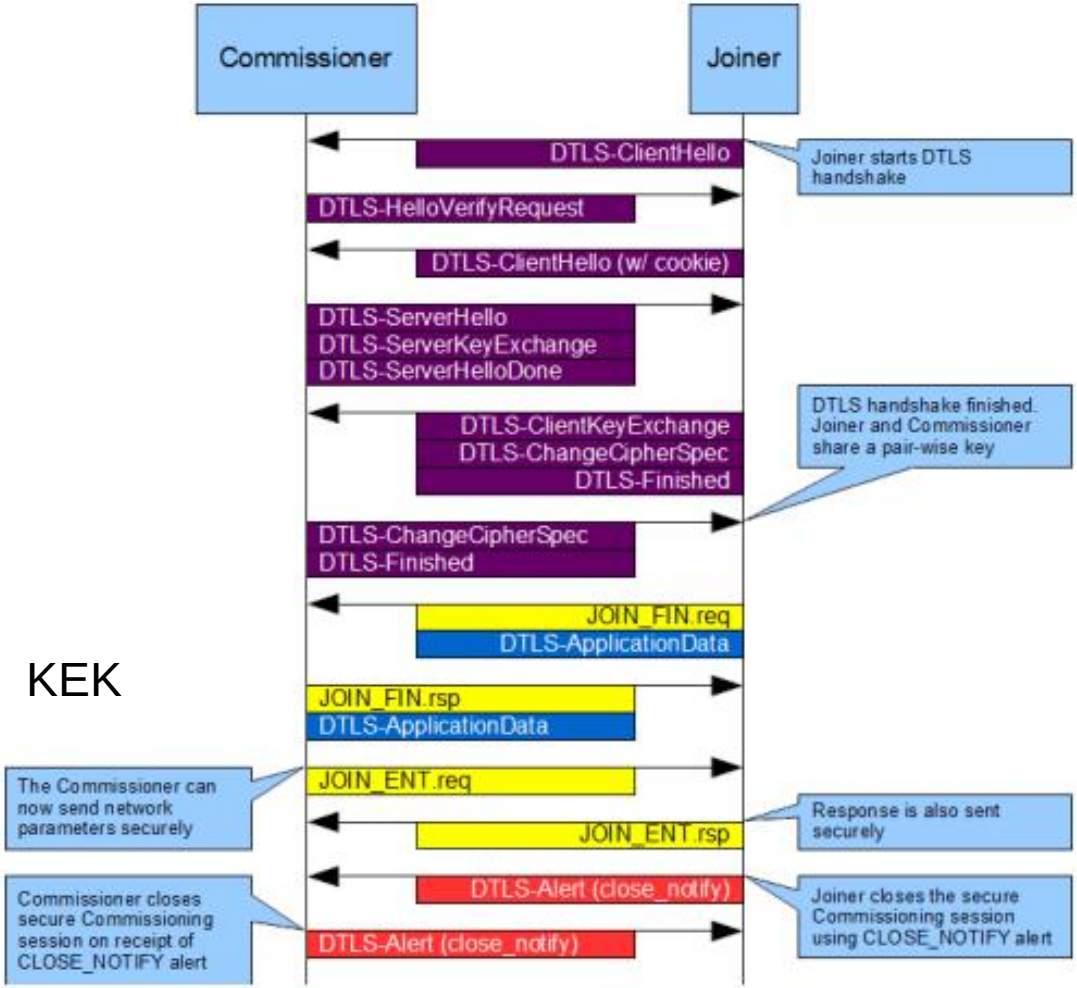


Case 3: Native Commissioner connected to the Thread Network, Joiner Router is not Commissioner

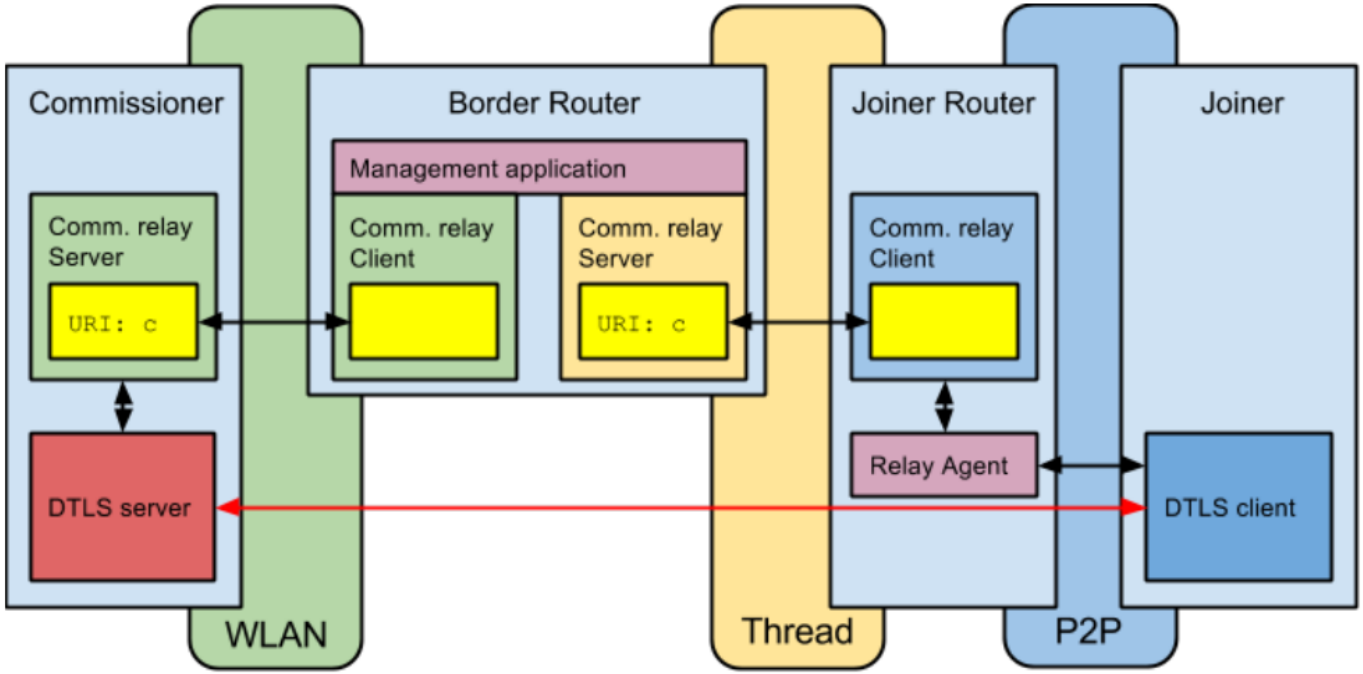


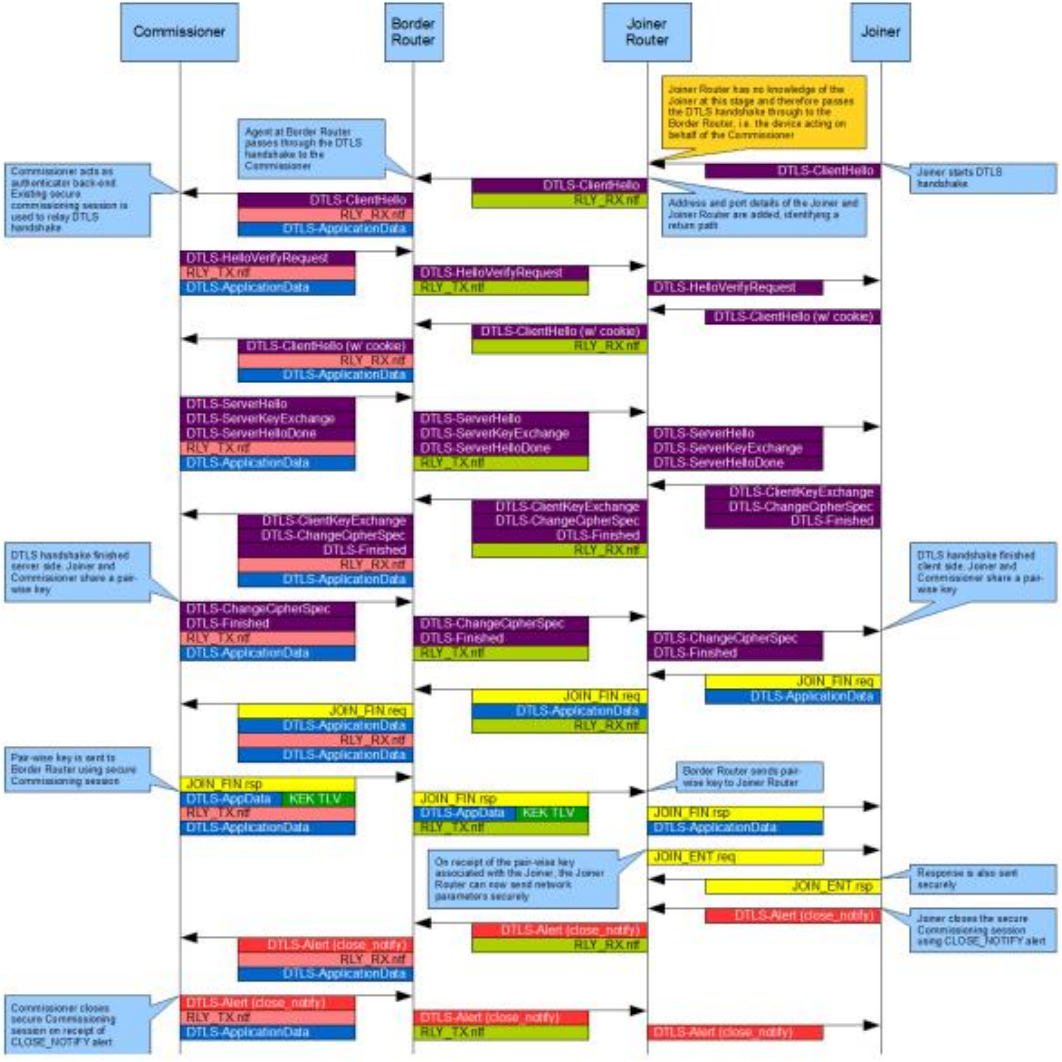
Joiner Router Is Commissioner



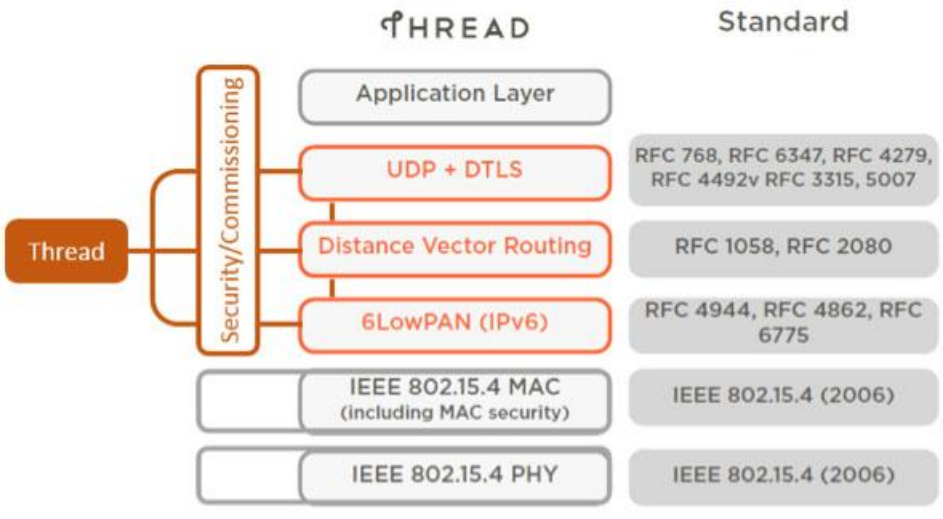

Commissioner
Joiner
DTLS-ClientHello
Joiner starts DTLS handshake
DTLS-HelloVerifyRequest
DTLS-ClientHello (w/ cookie)
DTLS-ServerHello
DTLS-ServerKeyExchange
DTLS-ServerHelloDone
DTLS-ClientKeyExchange
DTLS-ChangeCipherSpec
DTLS-Finished
DTLS handshake finished. Joiner and Commissioner share a pair-wise key
DTLS-ChangeCipherSpec
DTLS-Finished
JOIN_FIN.req
DTLS-ApplicationData
KEK
JOIN_FIN.rsp
DTLS-ApplicationData
The Commissioner can now send network parameters securely
JOIN_ENT.req
JOIN_ENT.rsp
Response is also sent securely
DTLS-Alert (close_notify)
Commissioner closes secure Commissioning session on receipt of CLOSE_NOTIFY alert
DTLS-Alert (close_notify)
Joiner closes the secure Commissioning session using CLOSE_NOTIFY alert

Joiner–Joiner Router–Border Router–Commissioner



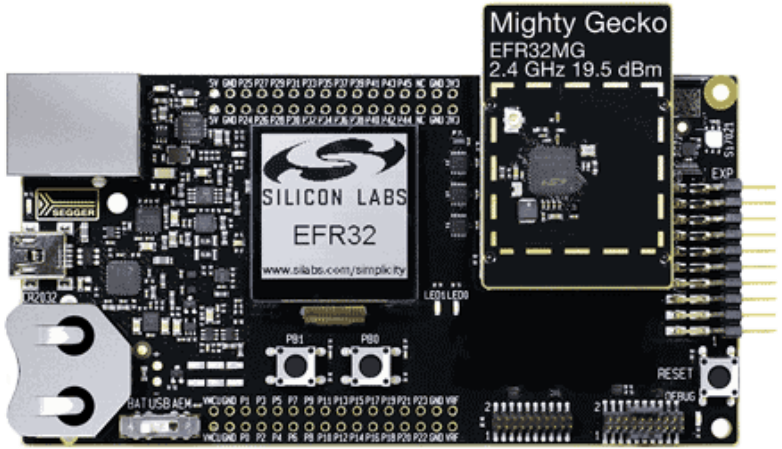

Commissioner
Border Router
Joiner Router
Joiner
Joiner Router has no knowledge of the Joiner at this stage and therefore passes the DTLS handshake through to the Border Router, i.e. the device acting on behalf of the Commissioner
Agent at Border Router passes through the DTLS handshake to the Commissioner
Commissioner acts as authenticator back-end. Existing secure commissioning session is used to relay DTLS handshake
DTLS-ClientHello
Joiner starts DTLS handshake
Address and port details of the Joiner and Joiner Router are added, identifying a return path
DTLS-ClientHello
RLY_RX.ntf
DTLS-ApplicationData
DTLS-HelloVerifyRequest
RLY_TX.ntf
DTLS-ClientHello (w/ cookie)
DTLS-ServerHello
DTLS-ServerKeyExchange
DTLS-ServerHelloDone
DTLS-ClientKeyExchange
DTLS-ChangeCipherSpec
DTLS-Finished
DTLS handshake finished server side. Joiner and Commissioner share a pair-wise key
DTLS handshake finished client side. Joiner and Commissioner share a pair-wise key
JOIN_FIN.req
JOIN_FIN.rsp
DTLS-AppData
KEK TLV
Pair-wise key is sent to Border Router using secure Commissioning session
Border Router sends pair-wise key to Joiner Router
JOIN_ENT.req
JOIN_ENT.rsp
On receipt of the pair-wise key associated with the Joiner, the Joiner Router can now send network parameters securely
Response is also sent securely
DTLS-Alert (close_notify)
Joiner closes the secure Commissioning session using CLOSE_NOTIFY alert
Commissioner closes secure Commissioning session on receipt of CLOSE_NOTIFY alert

THREAD BOARD

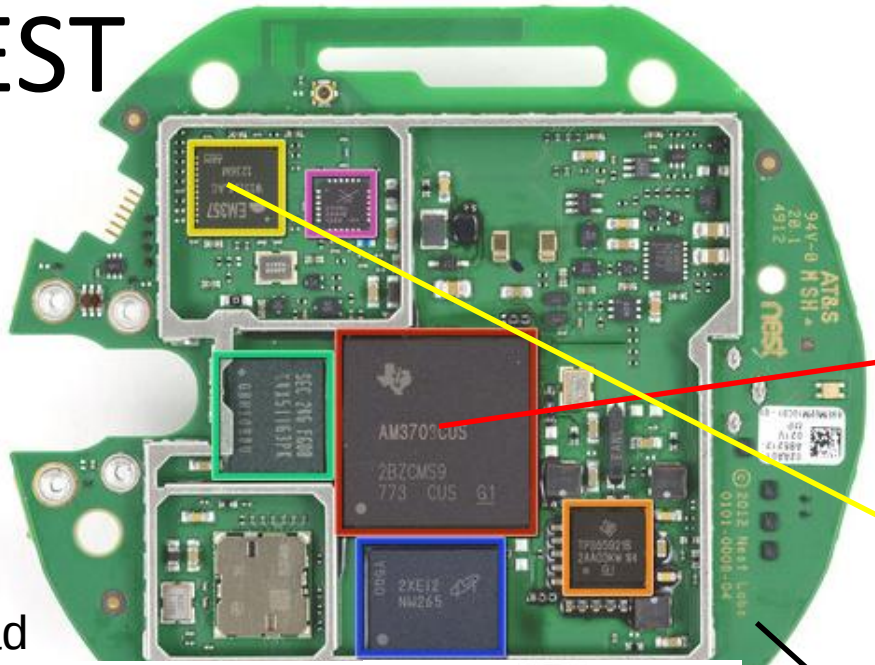


SILICON LABS

http://www.silabs.com/



NEST

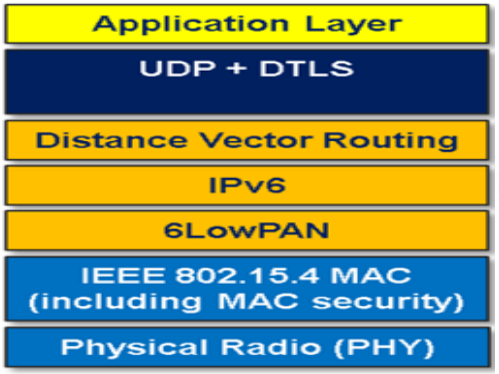


Step 15

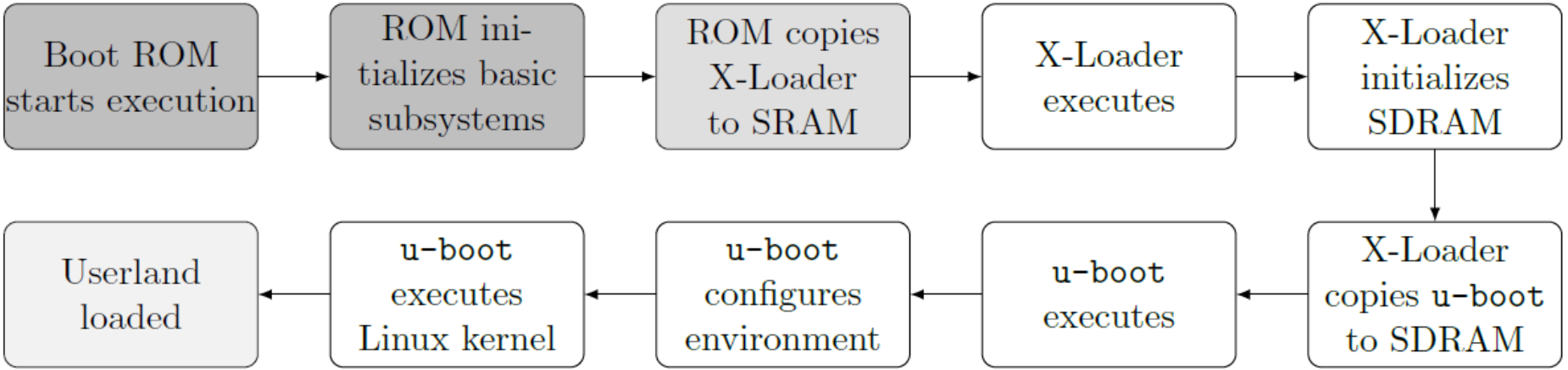
Edit

- With all of the I/O connections on the back, the main motherboard houses all of its important ICs on the front:
  - Texas Instruments AM3703CUS Sitara ARM Cortex A8 microprocessor
  - Texas Instruments TPS65921B power management and USB single chip
  - Samsung K4X51163PK 512 Mb mobile DRAM
  - Ember EM357 integrated ZigBee/802.15.4 system-on-chip
  - Micron MT29F2G16ABBEAH4 2 Gb NAND flash memory
  - Skyworks 2436L high power 2.4 GHz 802.15.4 front-end module
  - And under that last EMI shield: Texas Instruments WL1270B 802.11 b/g/n Wi-Fi solution, just like the one we found in the Kindle Fire

Thread



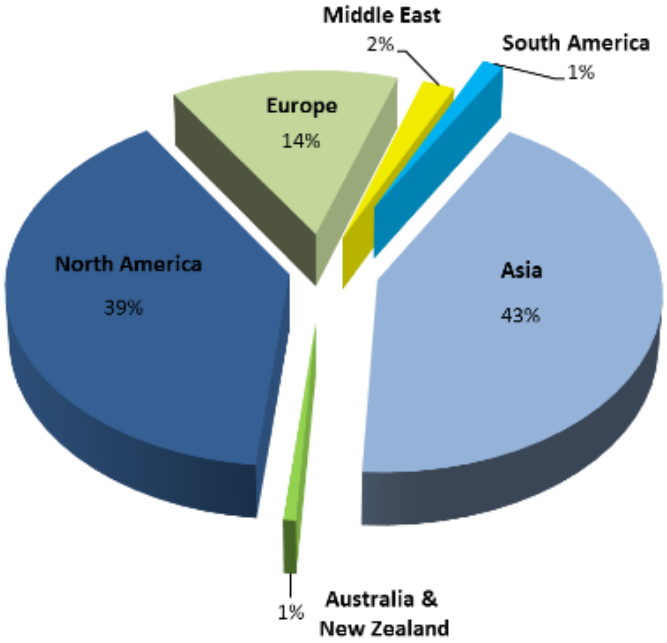
ascal Urien



A global reset of the device can be triggered by pressing its button for about 10 seconds. Among other things, this causes the sys boot5 pin to go high, triggering peripheral booting. Coincidentally, the sys boot5 pin is directly exposed in an unpopulated header within the main circuit board, which can be utilized to directly trigger the USB booting behavior. Since the ROM does no cryptographic checks of the code being loaded, it freely executes this code, allowing total control of the device.

"Smart Nest Thermostat: A Smart Spy in Your Home", Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin

Example 2.
Open Connectivity Foundation (OCF)



https://openconnectivity.org/

The **Open Connectivity Foundation (OCF)** is creating a specification and sponsoring an open source project to make this possible.
The OCF sponsors the IoTivity open source project which includes a reference implementation of our specification available under the Apache 2.0 license.

The OCF sponsors the IoTivity open source project which includes a reference implementation of our specification

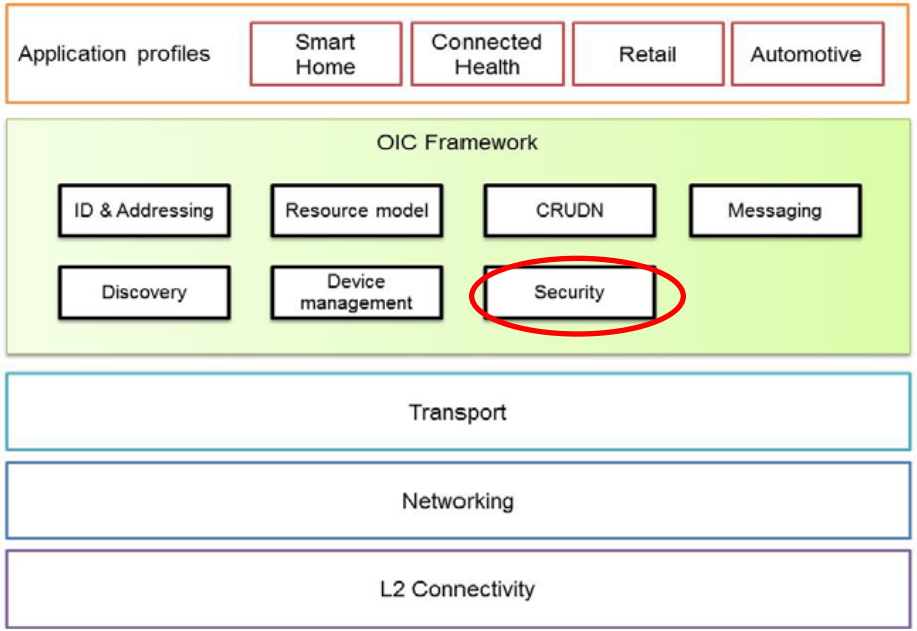
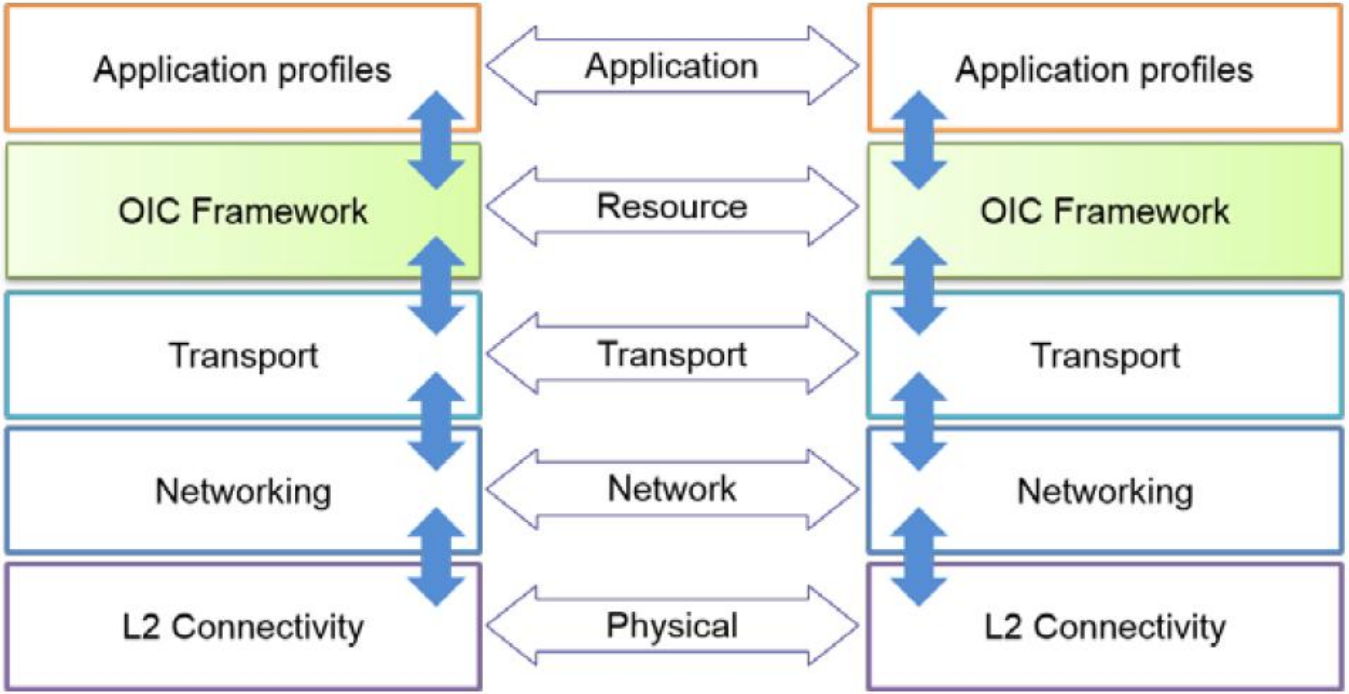


Figure 2: OIC functional block diagram

CRUDN: Create, Read, Update, Delete, Notify OIC: Open Interconnect Consortium

OCF Stack



OIC: Open Interconnect Consortium

- L2 connectivity: Provides the functionalities required for establishing physical and data link layer connections (e.g., Wi-Fi™ or Bluetooth® connection) to the network.

- Networking: Provides functionalities required for Devices to exchange data among themselves over the network (e.g., Internet).

- Transport: Provides end-to-end flow transport with specific QoS constraints. Examples of a transport protocol include TCP and UDP or new Transport protocols under development in the IETF, e.g., Delay Tolerant Networking (DTN).

- OIC Framework: Provides the core functionalities as defined in this specification. The functional block is the source of requests and responses that are the content of the communication between two Devices.

- Application profile: Provides market segment specific data model and functionalities, e.g., smart home data model and functions for the smart home market segment.

Security

- Secure Storage
  - It is strongly recommended that IoT device makers provide reasonable protection for Sensitive Data so that it cannot be accessed by unauthorized devices, groups or individuals for either malicious or benign purposes.
  - In addition, since Sensitive Data is often used for authentication and encryption, it must maintain its integrity against intentional or accidental alteration

Security

- Device Authentication with DTLS
  - Device Authentication with Symmetric Key Credentials
  - Device Authentication with Raw Asymmetric Key Credentials
  - Device Authentication with Certificates

Security

- Secure Boot
  - In order to ensure that all components of a device are operating properly and have not been tampered with, it is best to ensure that the device is booted properly.
  - There may be multiple stages of boot.
  - The end result is an application running on top an operating system that takes advantage of memory, CPU and peripherals through drivers.

Access Control List (ACL)

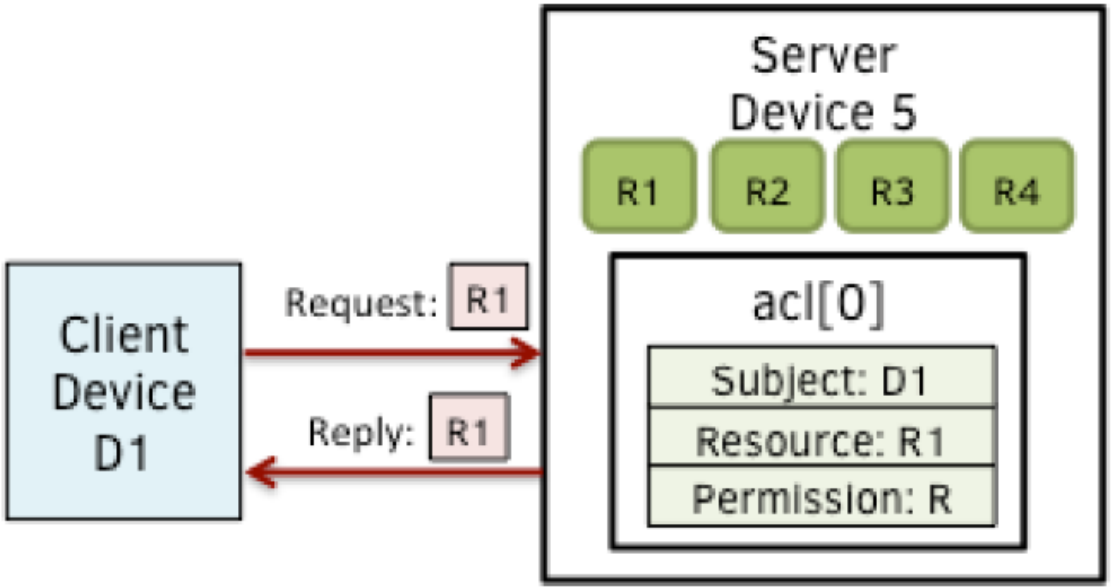
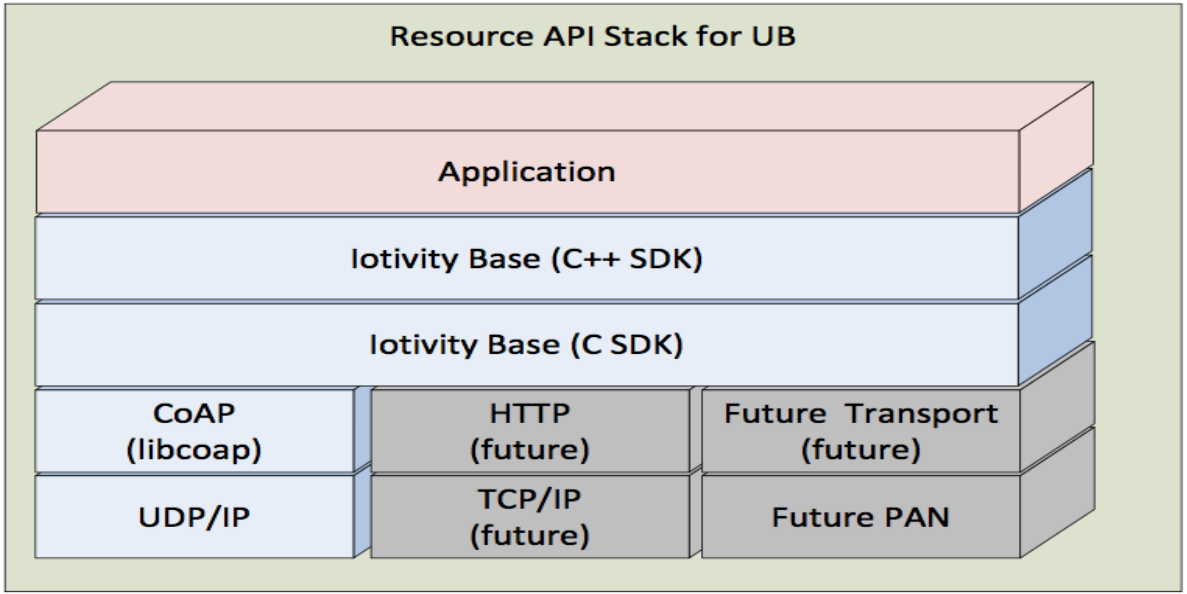


Figure 2 – Use case-1 showing simple ACL enforcement

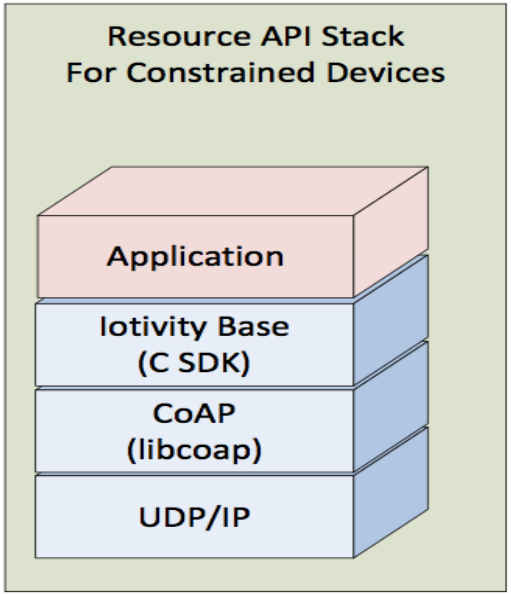
IOTIVITY

https://www.iotivity.org/

Unified Block (UB) stack



Thin Block (TB) stack



IoTivity is an open source software framework enabling seamless device-to-device connectivity to address the emerging needs of the Internet of Things.
It supports multiple operating systems : Linux, Android, Tize, Arduino

Unified Resource Identifier

oic://<Authority>/<Path>?<Query>

The usual form of the authority is :
<host>:<port>, where <host> is the name or endpoint network address and <port> is the network port number.

The *path shall be unique string that unambiguously identifies or references a resource within the context of the Server*

A query string shall contain a list of <name>=<value> segments (aka “name-value pair”) each separated by a ‘;’ (semicolon). The query string will be mapped to the appropriate syntax of the protocol used for messaging. (e.g., CoAP).

Resource = URI + Properties

```
/my/resource/example
{
"rt": "oic.r.foobar",
"if": "oic.if.a",
"value": "foo value"
}
```

URI

Properties

Resource Type ("rt")

Resource Interface ("if")

Resource Name ("n")

Resource Identity ("id"):

Properties are "key=value" pairs and represent state of the Resource

Interface	Name	Applicable Methods	Description
baseline	oic.if.baseline	RETRIEVE, UPDATE	The baseline Interface defines a view into all Properties of a Resource including the Meta Properties. This Interface is used to operate on the full Representation of a Resource.
links list	oic.if.ll	RETRIEVE	The 'links list' Interface provides a view into Links in a Collection (Resource). Since Links represent relationships to other Resources, the links list interfaces may be used to discover Resources with respect to a context. The discovery is done by retrieving Links to these Resources. For example: the Core Resource /oic/res uses this Interface to allow discovery of Resource "hosted" on a Device.
batch	oic.if.b	RETRIEVE, UPDATE	The batch Interface is used to interact with a collection of Resources at the same time. This also removes the need for the Client to first discover the Resources it is manipulating – the Server forwards the requests and aggregates the responses
read-only	oic.if.r	RETRIEVE	The read-only Interface exposes the Properties of a Resource that may be 'read'. This Interface does not provide methods to update Properties or a Resource and so can only be used to 'read' Property Values.
read-write	oic.if.rw	RETRIEVE, UPDATE	The read-write Interface exposes only those Properties that may be both 'read' and "written" and provides methods to read and write the Properties of a Resource.
actuator	oic.if.a	CREATE, RETRIEVE, UPDATE	The actuator Interface is used to read or write the Properties of an actuator Resource.
sensor	oic.if.s	RETRIEVE	The sensor Interface is used to read the Properties of a sensor Resource.

OCF REST

Request: GET /a/act/heater?if="oic.if.a"

Response:
{ "prm": {"sensitivity": 5, "units": "C",
"range": "0 .. 10"},
"settemp": 10,
"currenttemp" : 7
}

Request: POST /a/act/heater?if="oic.if.a "
{ "settemp": 20 }

Response:
{ Ok }

oic://server:port

```
/my/resource/example
{
"rt": "oic.r.foobar",
"if": "oic.if.a",
"value": "foo value"
}
```

URI

Properties

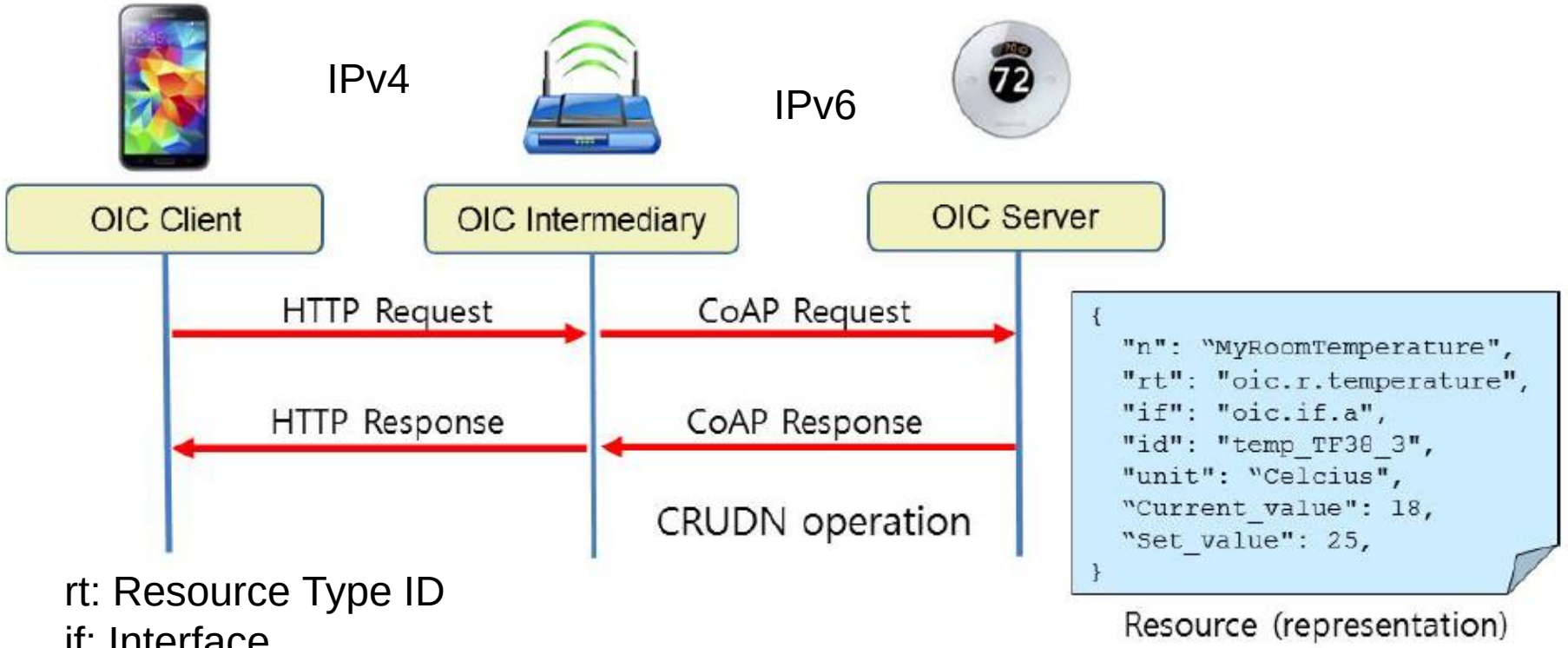
Resource Type ("rt")

(Resource) Interface ("if")

(Resource) Name ("n")

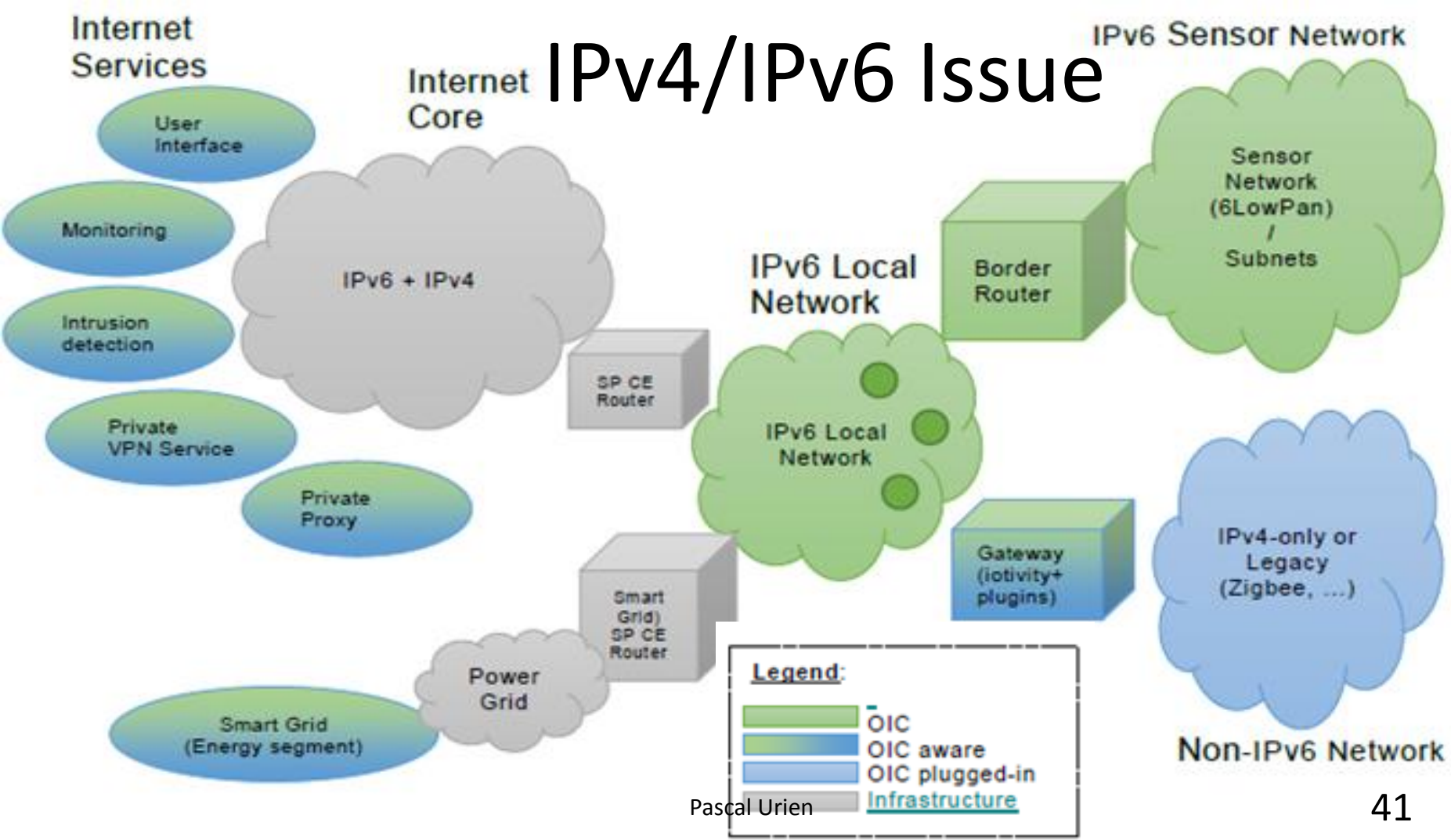
Resource Identity ("id"):

CoAP /HTTP



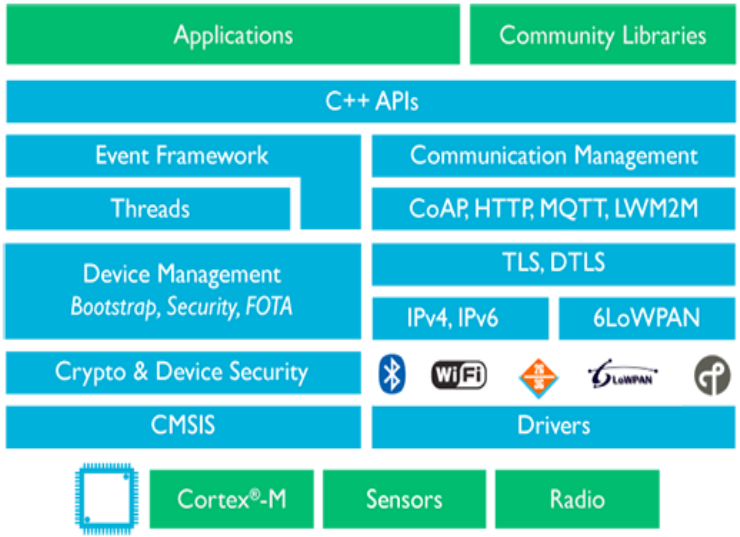
rt: Resource Type ID
if: Interface

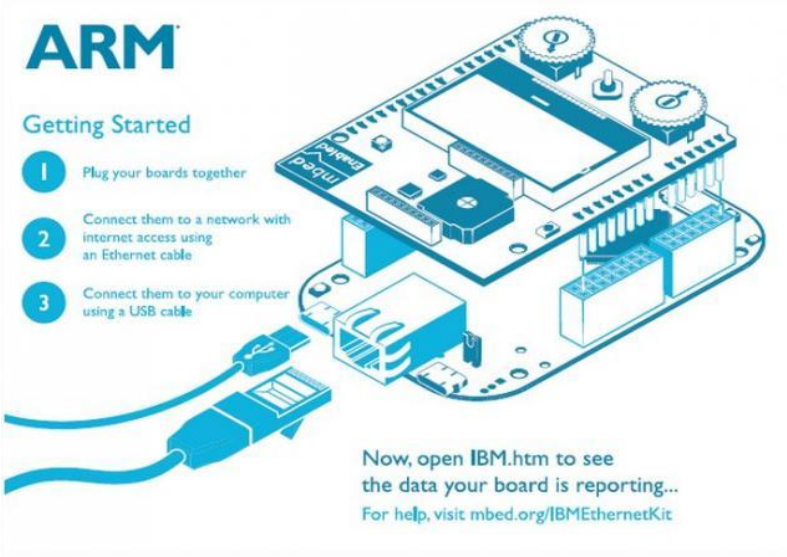
IPv4/IPv6 Issue



Example 3. MBED

MBED stack from the ARM company

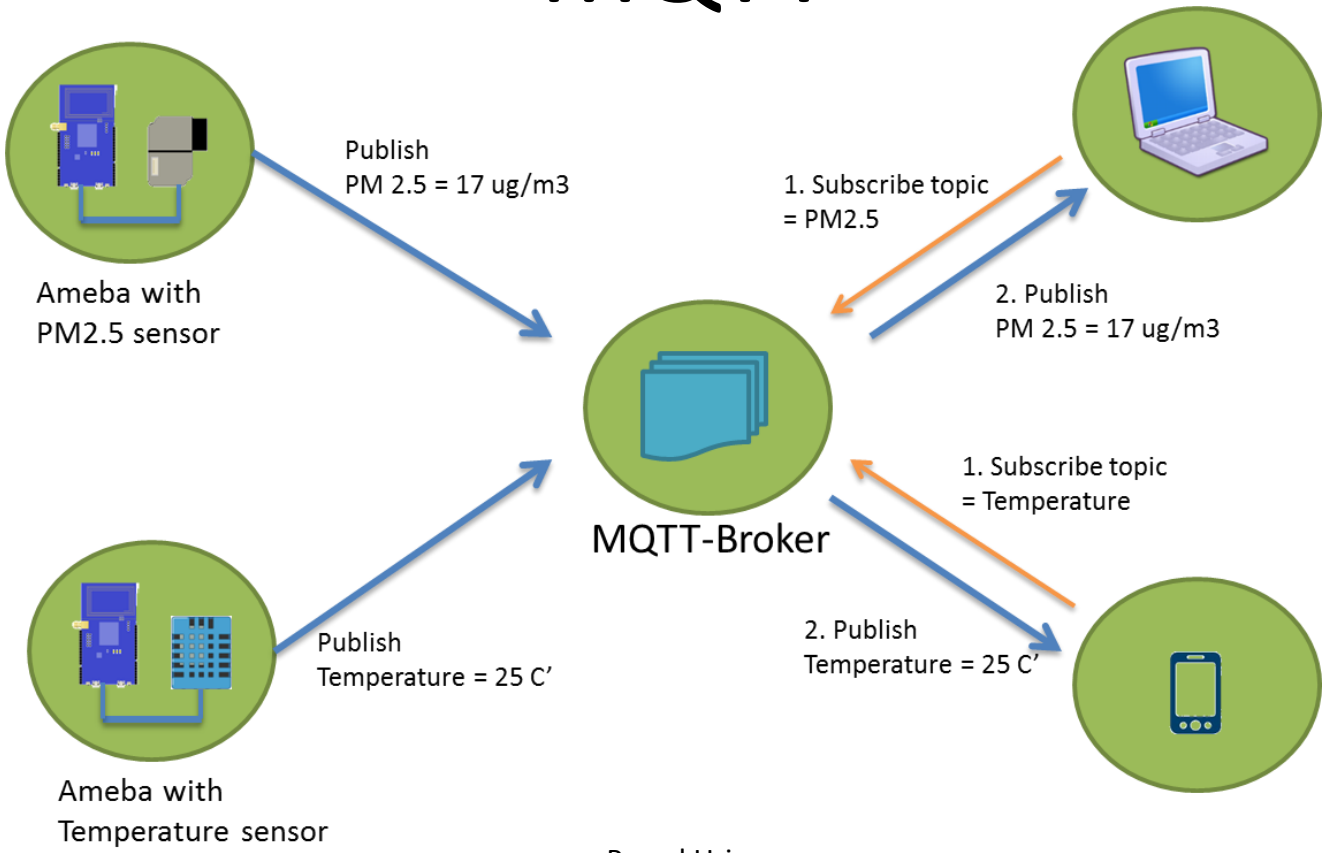




IoT Protocols

- HTTP (most of today IP objects)
  - As an illustration some connected plugs work with the HNAP (*Home Network Administration Protocol*) protocol based on SOAP and used in CISCO routers. In 2014 HNAP was infected by" The Moon".
- MQTT protocol, is a Client Server publish/subscribe messaging transport protocol that is secured by TLS.

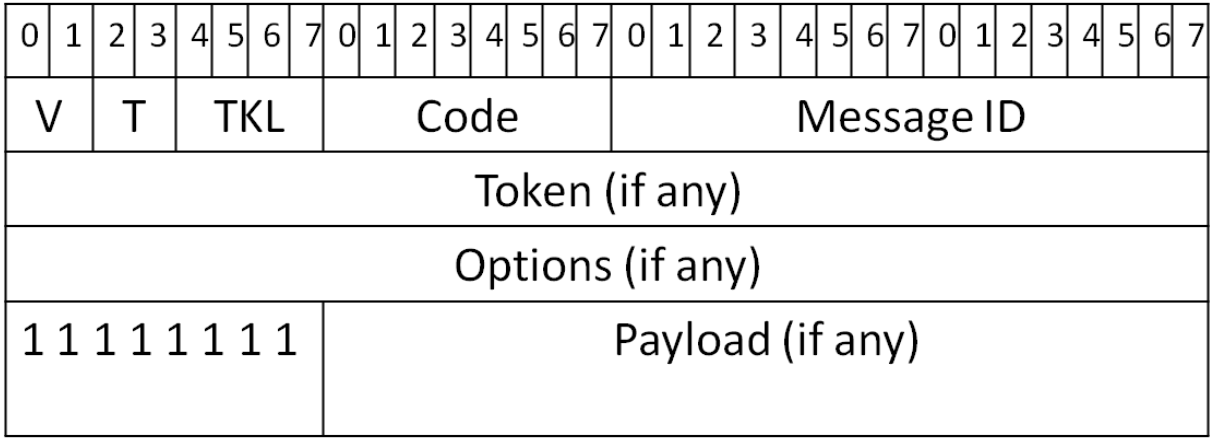
MQTT


Publish
PM 2.5 = 17 ug/m3
1. Subscribe topic
= PM2.5
2. Publish
PM 2.5 = 17 ug/m3
Ameba with
PM2.5 sensor
MQTT-Broker
1. Subscribe topic
= Temperature
2. Publish
Temperature = 25 C'
Publish
Temperature = 25 C'
Ameba with
Temperature sensor

CoAP, RFC 7252

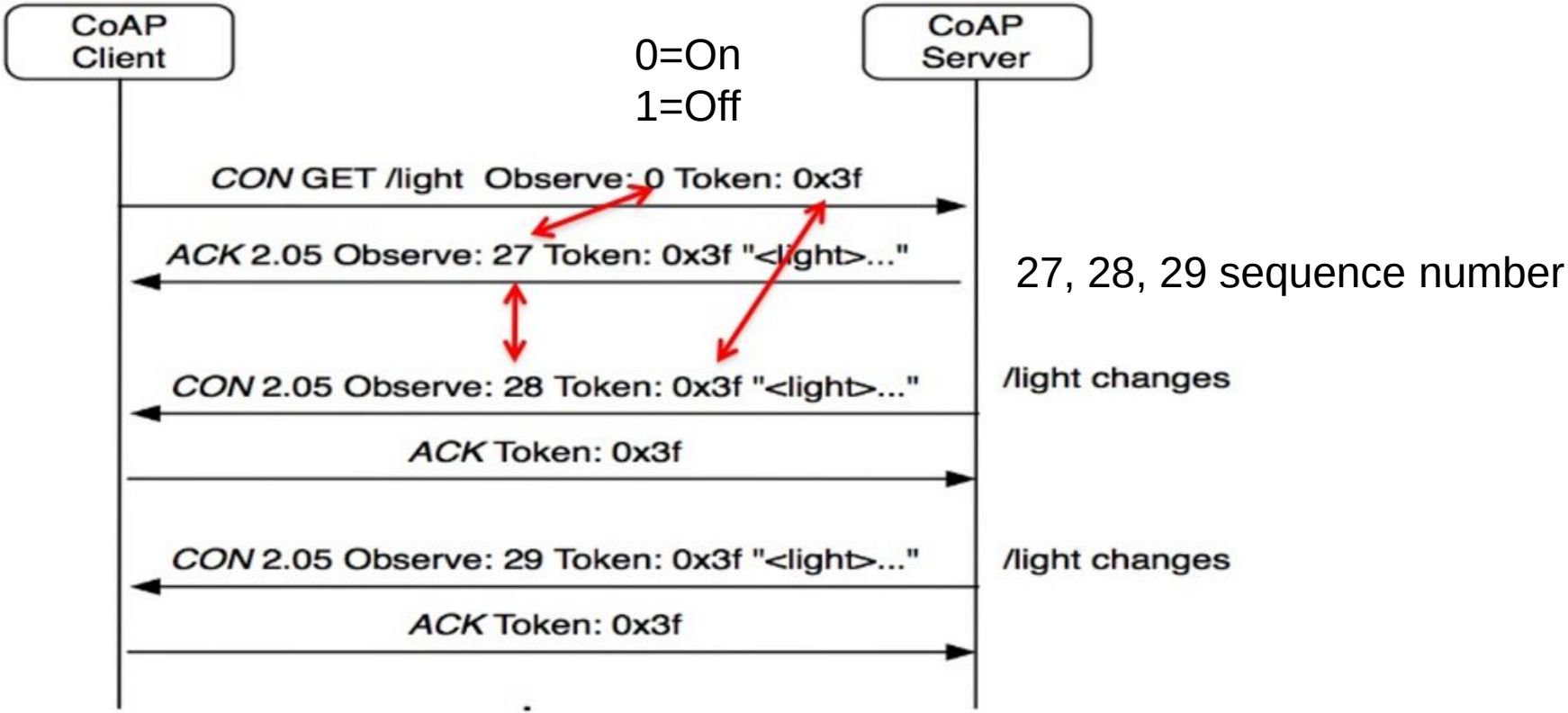
- CoAP (Constrained Application Protocol) , RFC 7252 is designed according to the Representational State Transfer (REST) architecture , which encompasses the following six features:
  - 1) Client-Server architecture;
  - 2) Stateless interaction;
  - 3) Cache operation on the client side;
  - 4) Uniform interface ;
  - 5) Layered system ;
  - 6) Code On Demand.
- CoAP is an efficient RESTfull protocol easy to proxy to/from HTTP, but which is not understood in an IoT context as a general replacement of HTTP.
  - It is natively secured by DTLS (the datagram adaptation of TLS), and works over a DTLS/UDP/IP stack. Nerveless the IETF is currently working on a CoAP version compatible with a TLS/TCP/IP stack.

CoAP Details

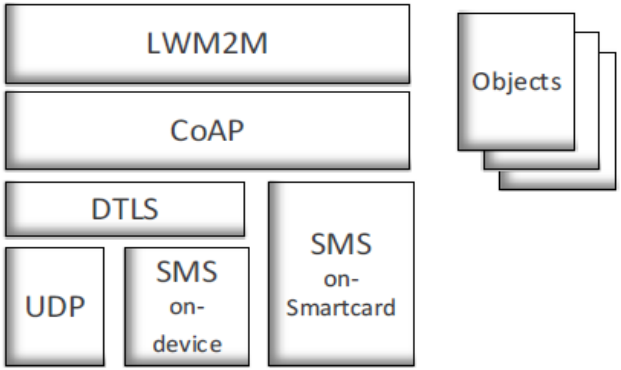


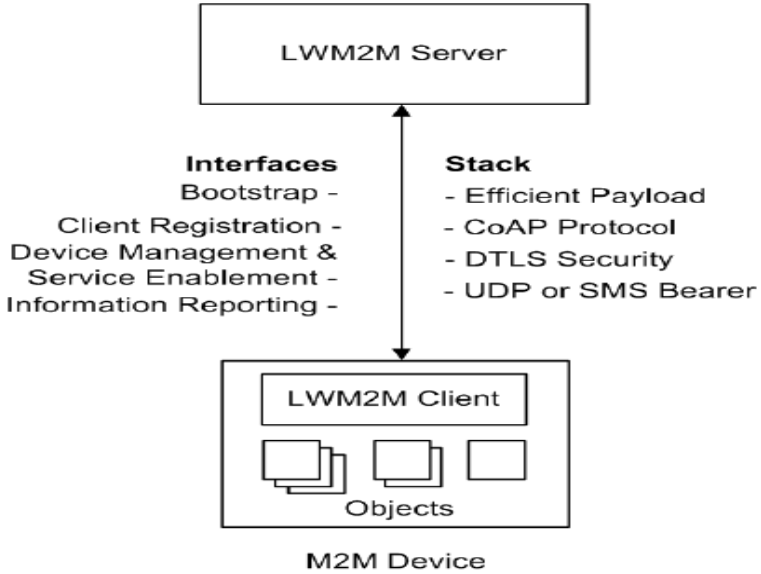
Version (V): protocol version (01).
Type (T) message type :
Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK) or Reset.
Token Length (TKL)/ is the length of the Token field (0-8 bytes).
The Code field: identifies the method and is split in two parts a 3-bit class and a 5-bit detail documented as "c.dd" where "c" is a digit from 0 to 7 and "dd" are two digits from 00 to 31.
0.01 GET, 0.02 POST, 0.03 PUT and 0.04 DELETE.
Message ID: matches messages ACK/Reset to messages CON/NON previously sent.
The Token (0 to 8 bytes): is used to match a response with a request.
Options: give additional information such as Content-Format dealing with proxy operations.

Observe option (Observe: int value)



LWM2M

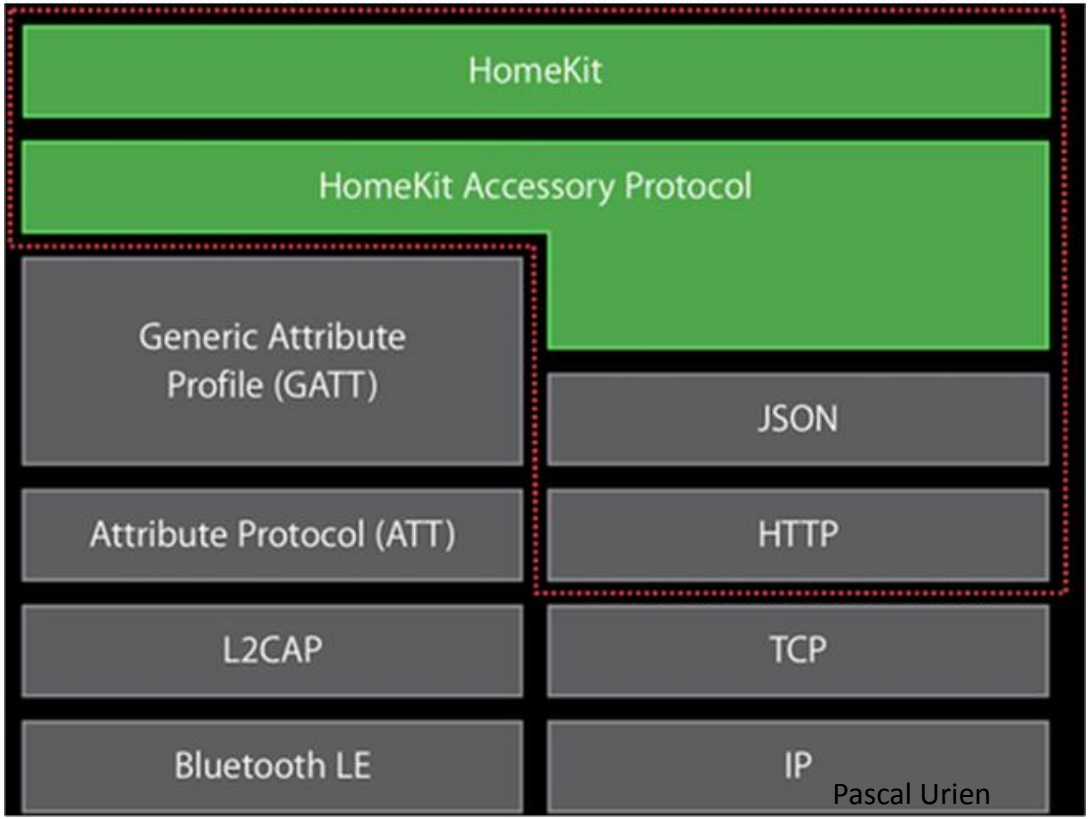




- LWM2M (*Lightweight Machine to Machine Technical Specification)* is a framework based on CoAP dealing with objects hosted by LWM2M clients and communicating with LWM2M servers
- LWM2M manages the following interfaces
  - Bootstrap
  - Client Registration (with servers)
  - Device management
  - Information Reporting
- Two transport mechanism ("*transport channel bindings“)*
  - *UDP/IP*
  - *SMS*

Example 4. Home Kit

HOME Kit (Apple)



Protocol Security
- End-to-end encryption
- Initial setup secured directly between iOS and accessory
- Perfect forward secrecy
- Standard cryptography

The HAP (*HomeKit Accessory Protocol*) initial pairing exchange is based on the Secure Remote Password procedure (SRP, RFC 5054) which deals with a 8 digits PIN code available for every accessory.

HAP Security Details

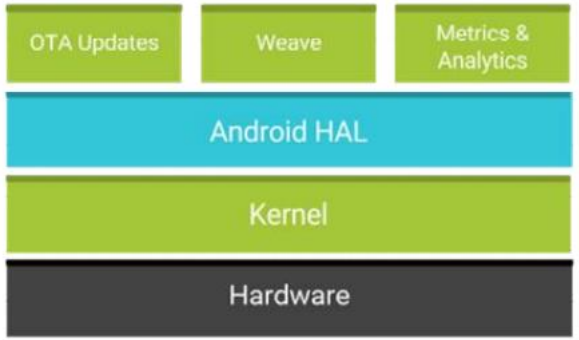
- Secure Remote Password (SRP) Encrypts and authenticates initial pairing key exchange
- Ed25519 Long-term keys for pairing and authentication
- Curve25519 Encrypts initial authentication for each session
- HKDF-SHA-512 Per-session ephemeral encryption key derivation
- ChaCha20-Poly1305 Encrypts and authenticates HAP data

Example 5. Brillo & Weave

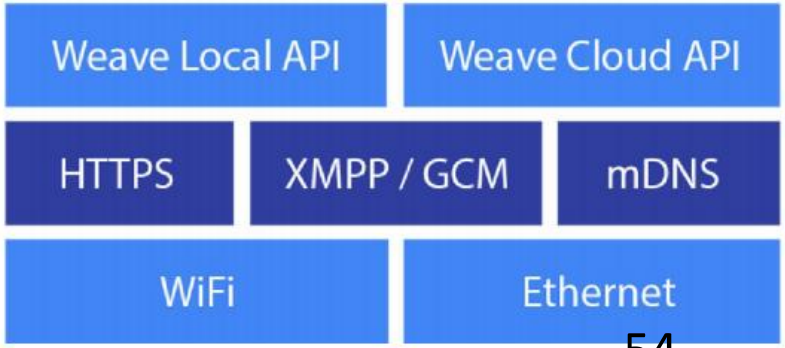
Brillo & Weave

The Intel® Edison Board Made for Brillo.

Brillo is an OS from Google for building connected devices. 35MB Memory Footprint (minimum)



Weave is a communications protocol that supports discovery, provisioning, and authentication so that devices can connect and interact with one another, the Internet, and your mobile platforms.

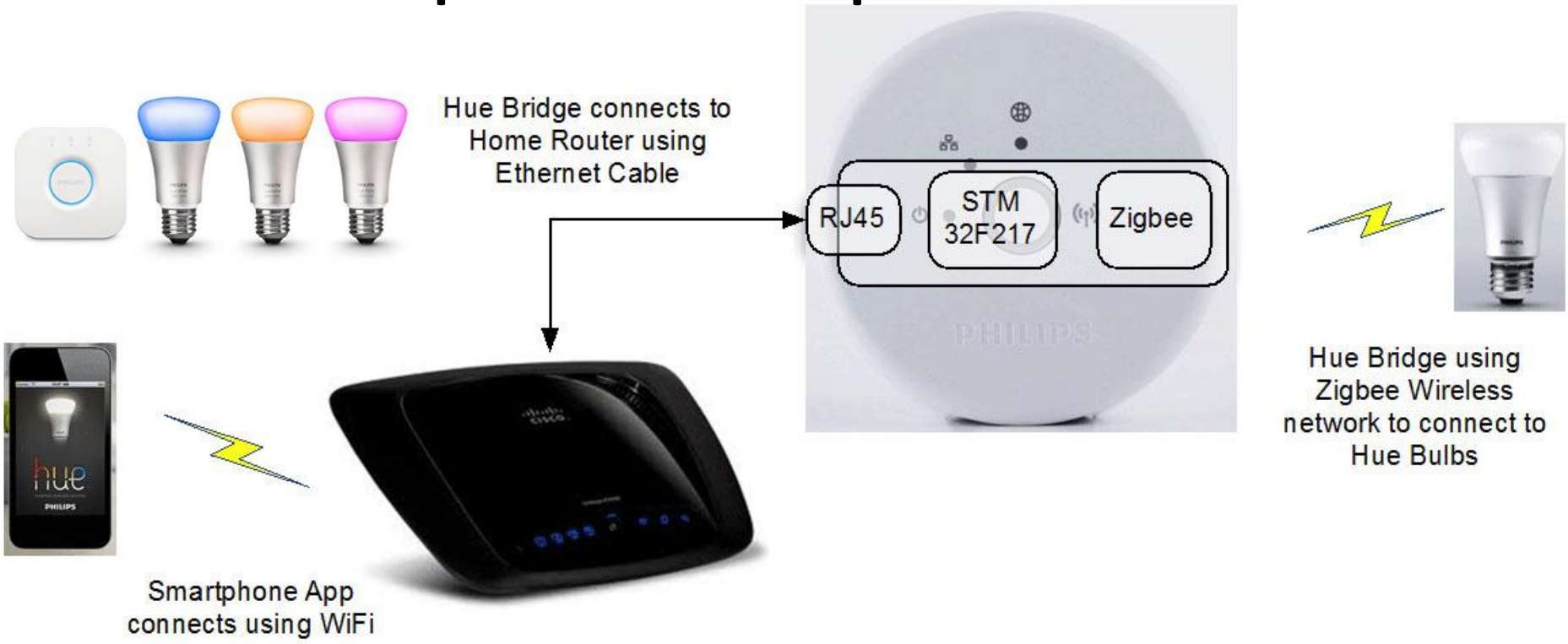


Brillo and Weave

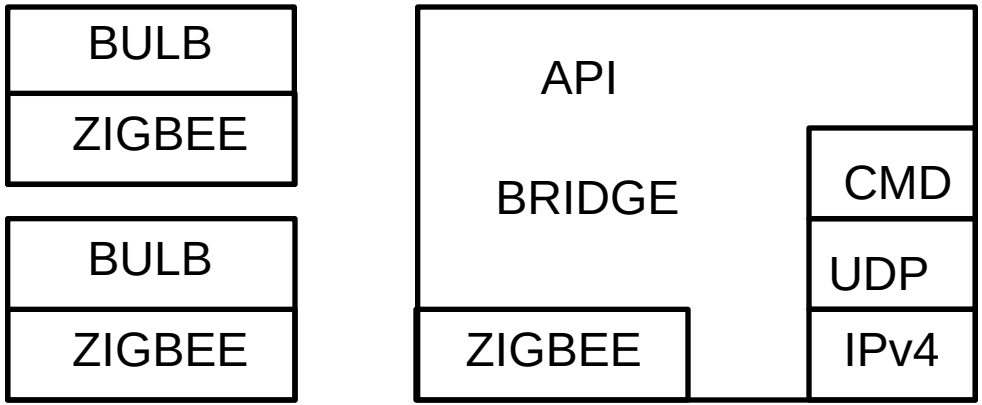
Weave is a communications platform for IoT devices
- Device setup, phone-to-device-to-cloud communication
- User interaction from mobile devices and the web
- Transports: 802.15.4 (zigbee, threads), BLE, Wi-Fi, Ethernet, Others possible
- Schema Driven (JSON) Associates Weave XMPP requests with application function invocations
 - Web apps may be written with Google API support
- OAuth 2.0 Authentication, Google as Authentication Server (AS)

Brillo is Simpler… Smaller…IoT Focused
- C/C++ environment
- Binder IPC No Java Applications, framework, runtime
-No Graphics
- 35MB Memory Footprint (minimum)

Example 6. Philips Hue Bulbs



Hue Bulb System



The commands are sent in the UDP payload as a short series of bytes with a termination byte of 0x55. For example, the light on command is:

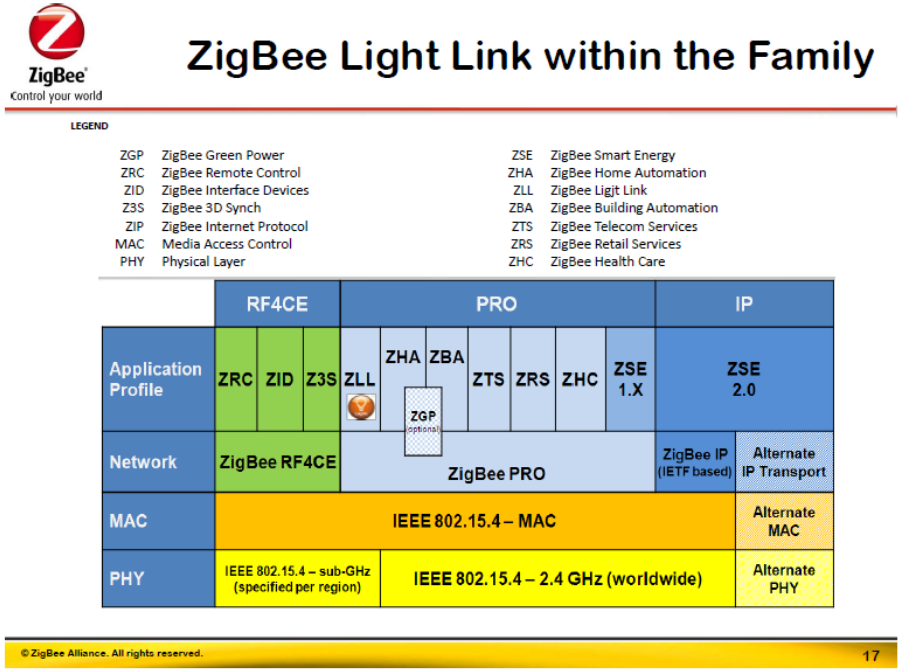
IPHEADER	UDPHEADER	0x45	0x00	0x55

and the light off command is:

IPHEADER	UDPHEADER	0x41	0x00	0x55

Extended Functionality Attacks on IoT Devices: The Case of Smart Lights (Invited Paper), Eyal Ronen, Adi Shamir

http://www.developers.meethue.com/



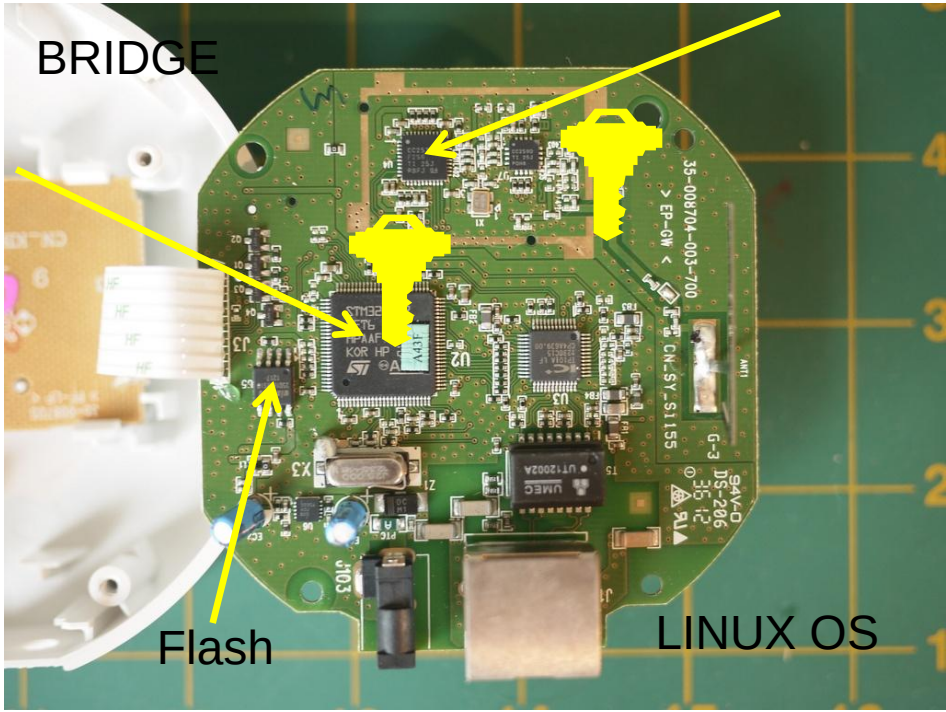
“The ZLL security architecture is based on using a fixed secret key, known as the ZLL key, which shall be stored in each ZLL device. All ZLL devices use the ZLL key to encrypt/decrypt the exchanged network key. “

https://brandonevans.ca/projects/hacking-the-hue

A LIGHTBULB WORM?, Details of the Philips Hue Smart Lighting Design, Colin O'Flynn – August 1, 2016.

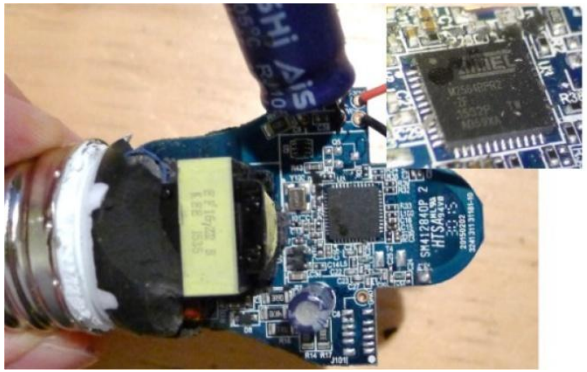
ZigBee SoC Includes a Hardware Accelerator)

These bridges contain two sections: the main ARM processor, and the Zigbee ZLL solution (referred to as the 'Zigbee SoC').
The main ARM processor is a STM32F217VET6 by ST.
This is a Cortex M3 device, with 512 Kbyte FLASH memory (internal) + 128 Kbyte of SRAM (internal). It contains a number of cryptographic hardware accelerators (AES + 3DES + MD5 + SHA-1).

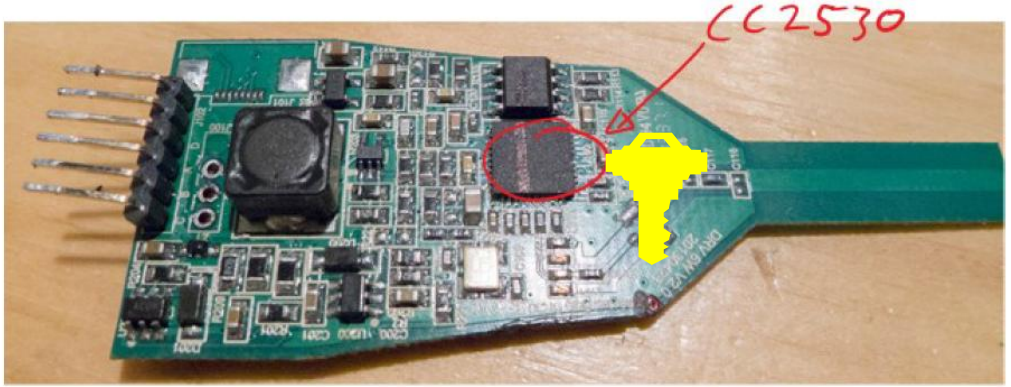


The ZigBee section is of most interest to us. It contains a CC2530F256 IEEE 802.15.4 SoC, alongside a CC2590 "range extender" (i.e., amplifier)

The core processor is an Atmel ATMega2564RFR2.



The firmware updates are downloaded Over The Air (OTA).
The firmware file itself can be downloaded from a fixed URL, and contains an encrypted firmware file (similar to the firmware update for the CC2530 device).

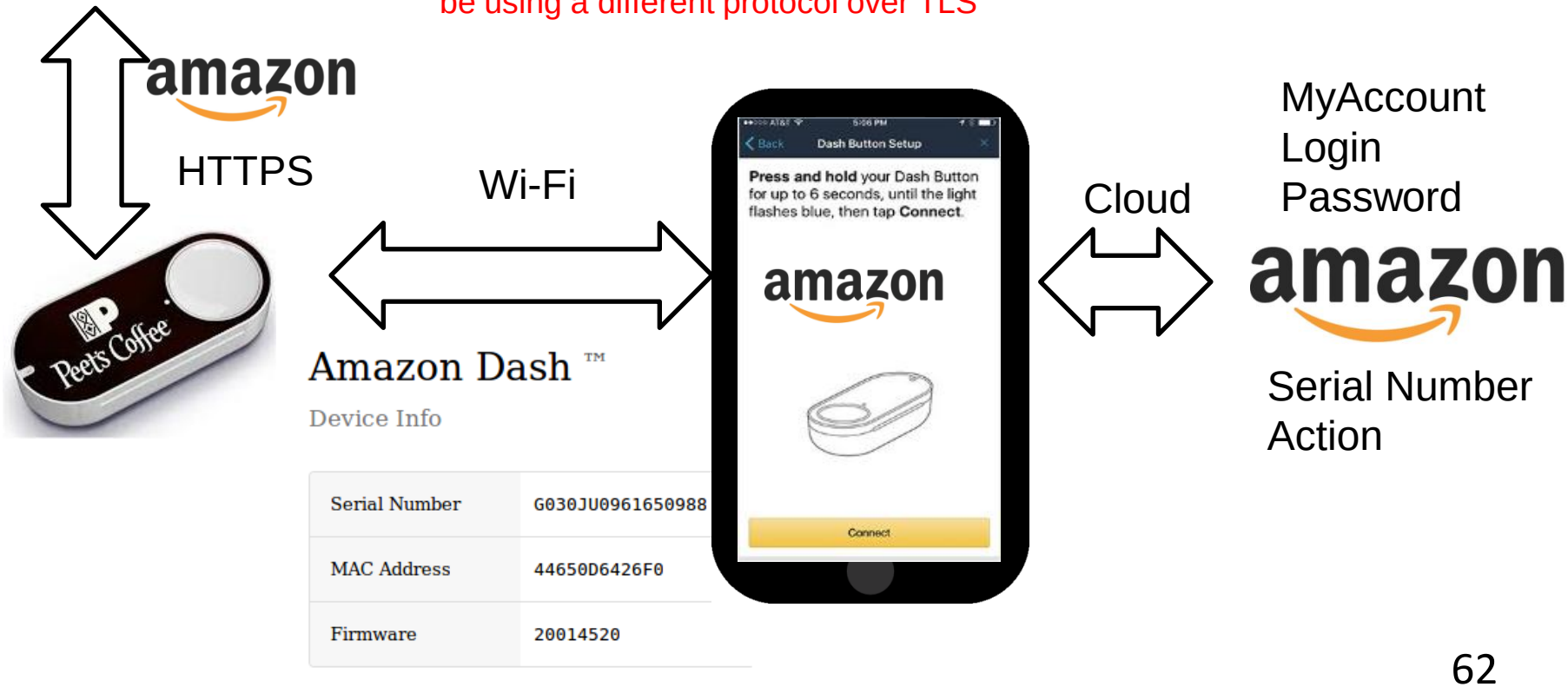


https://plus.google.com/photos/107696725527584609973/albums/5806291983792940817

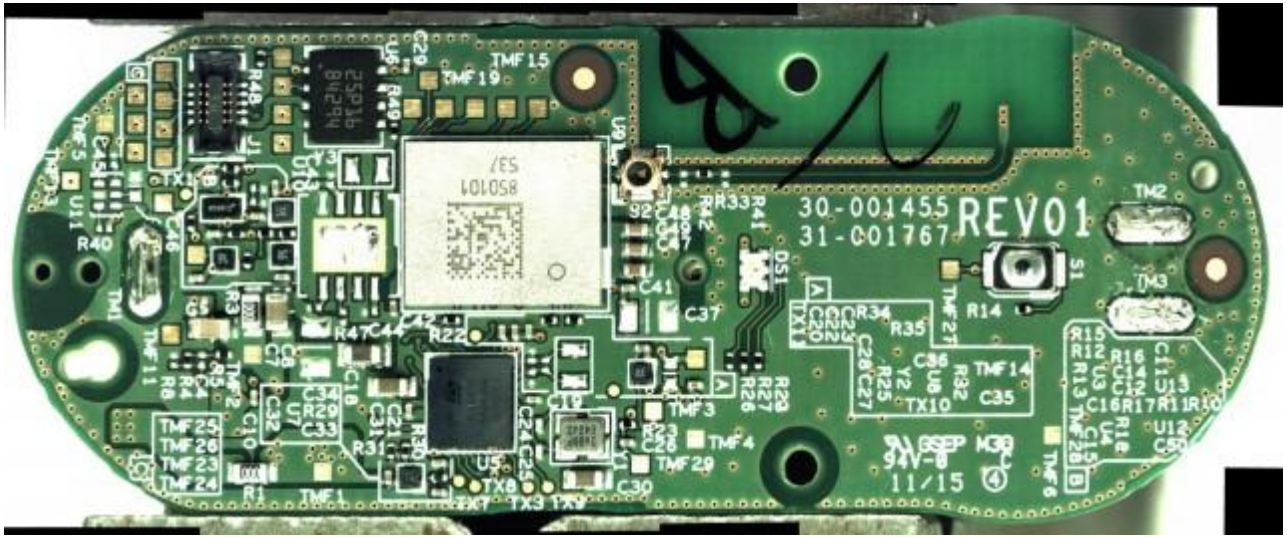
Example 7.
Amazon Dash Button

Button communicates with parker-gateway-na.amazon.com via TLS

When connecting via HTTPS, a certificate signed by the Amazon.com Internal Root Certificate Authority and issued to Amazon.com Infosec CA G2 is presented, which expires 2016-06-22. However, I was not able to successfully connect even after bypassing the certificate error, so it might be using a different protocol over TLS



https://mpetroff.net/2015/05/amazon-dash-button-teardown/



The design seems based on the Broadcom BCM943362WCD4 WICED module reference design, with a Broadcom BCM43362 Wi-Fi module, U9, and an ST STM32F205 microcontroller, U5

Other components on the Dash Button include an InvenSense INMP441 microphone, MP1; a Micron M25P16 16Mbit serial Flash memory module in a UFDFPN8 package, U6

Although not mentioned in the documentation, the Dash Button creates a Wi-Fi hotspot when placed in configuration mode, Amazon ConfigureMe, which is used by the Android version of the Amazon Shopping app.
Once connected to this hotspot, a web page is accessible at 192.168.0.1 via HTTP, which allows for configuring the Button's Wi-Fi connection settings.