# Connect everything with CoAP


**Benjamin Cabé**
Eclipse Foundation

JavaOne™

# Follow the slides

https://goo.gl/U7kWKw

# Agenda

Internet of things 101

What protocols should I use?

CoAP

- What is CoAP?

CoAP live!

Californium

HANDS-ON!

More CoAP goodies

# What you will need

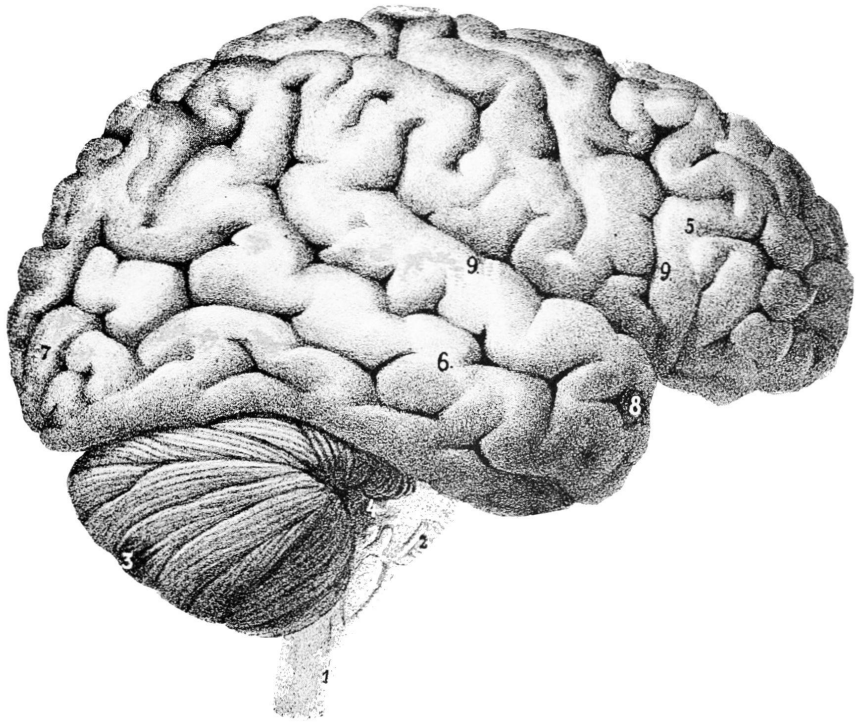
Eclipse IDE

Basic Java knowledge

Californium JARs

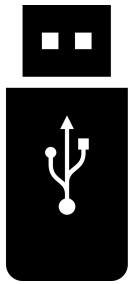
Firefox + Copper

Your brainzzz



# Contents of the USB stick

- Eclipse IDE for Windows, Linux and Mac
- Firefox and Copper .xpi
- Sample projects to be imported in your workspace
  + Californium JAR file
- Completed projects



# Machine to machine?

Machine to machine?
Internet of things?

Technology that supports wired or wireless communication between devices

# Different needs, different protocols

## Device Management

Radio statistics, device configuration, ...

OMA-DM, TR-069, LWM2M...

## Local sensor networks

Transmit sensor data, usually over RF or PLC

Zigbee, X10, Bluetooth Smart, ...

## End-user applications

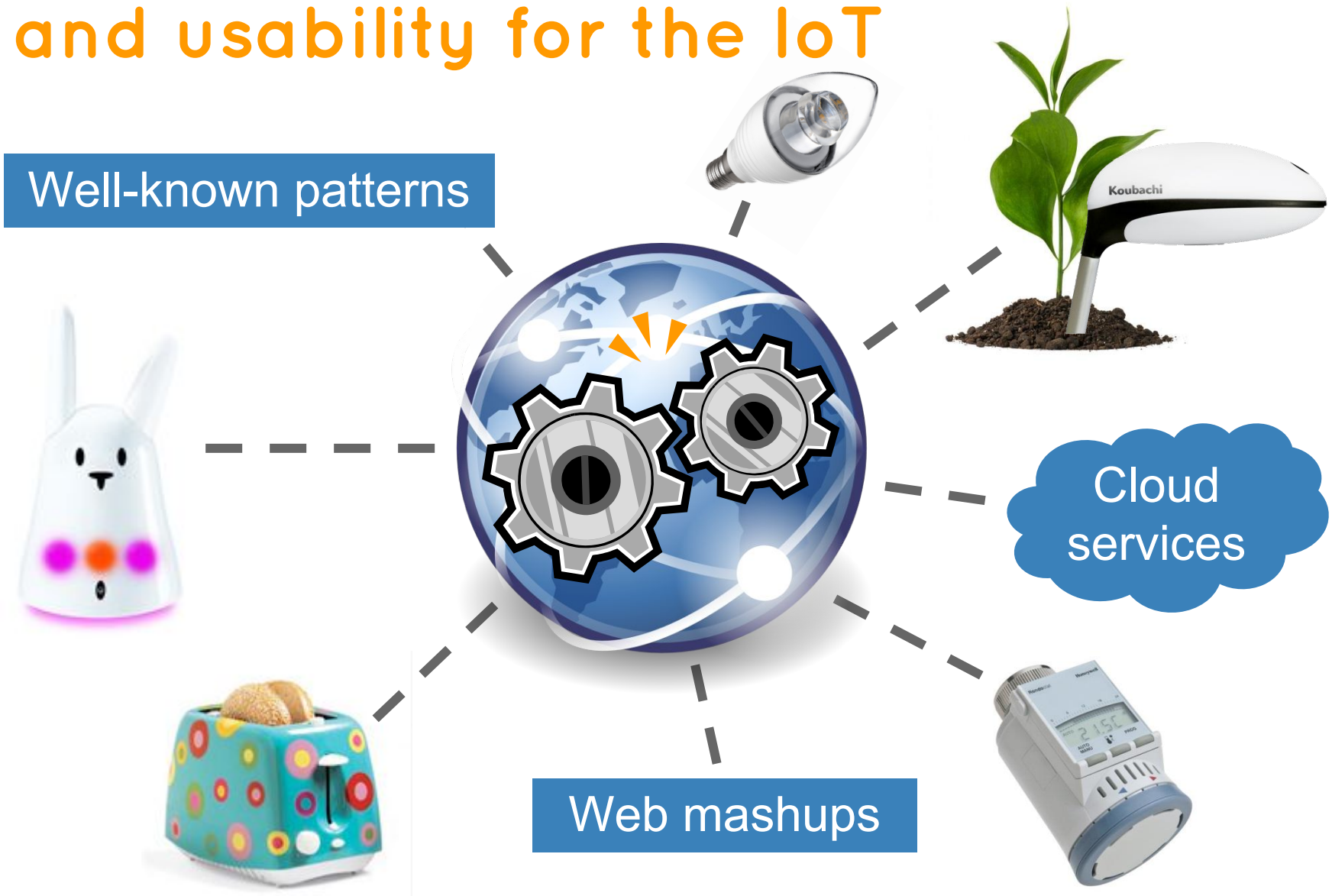
Display sensor data on mobile app, dashboards,

HTTP, Websockets, ...

# The Web of Things

Slide courtesy
of Vlad Trifa

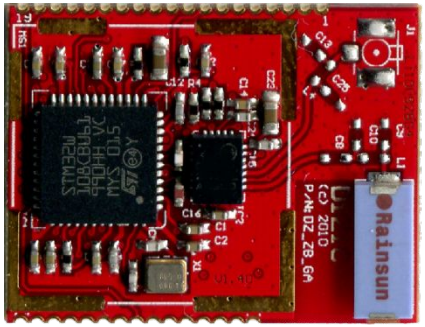
# Application-layer interoperability and usability for the IoT



# Tiny Resource-constrained devices

**Class 1 devices**
~100KiB Flash
~10KiB RAM



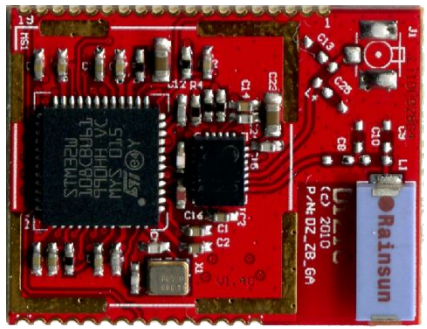



**Low-power networks**

# Tiny Resource-constrained devices

**Target**
of less than $1
for IoT SoC





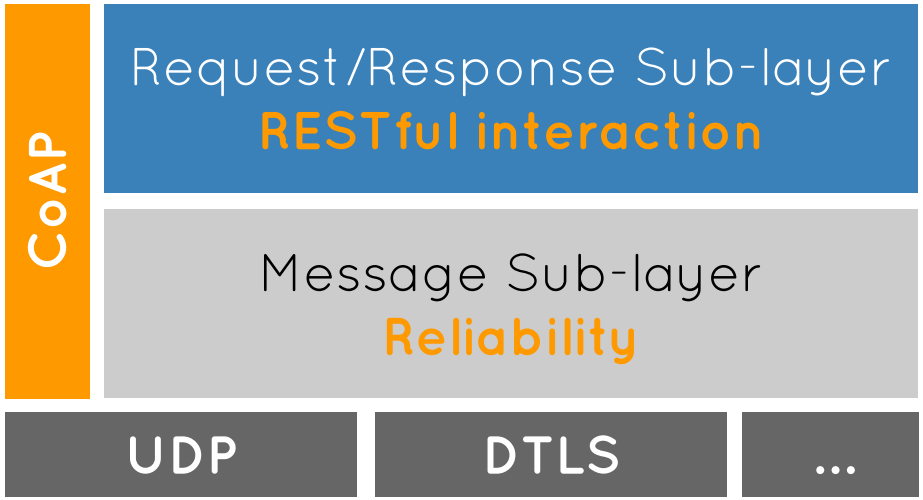

**TCP and HTTP**
are not a good fit

# Constrained Application Protocol

RESTful protocol designed from scratch

Transparent mapping to HTTP

Additional features for M2M scenarios




**GET**, **POST**, **PUT**, **DELETE**
URIs and Internet Media Types

Deduplication
Optional retransmissions
(Confirmables "CON")

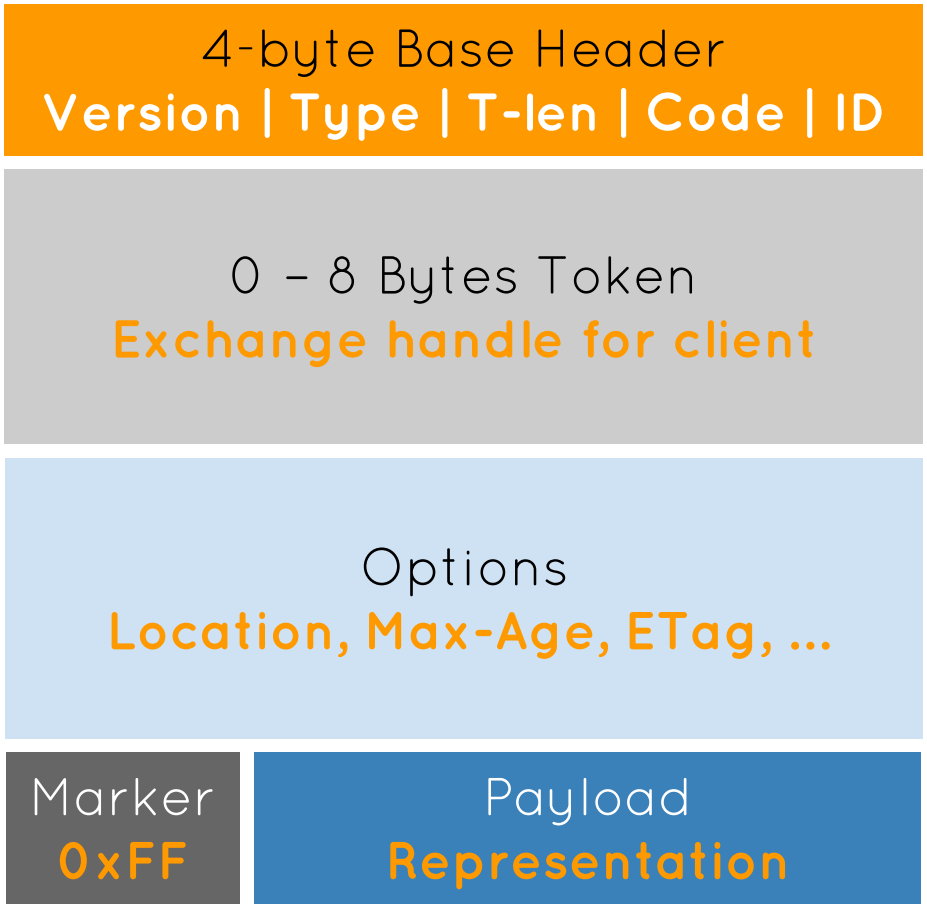
# Constrained Application Protocol

## Binary protocol

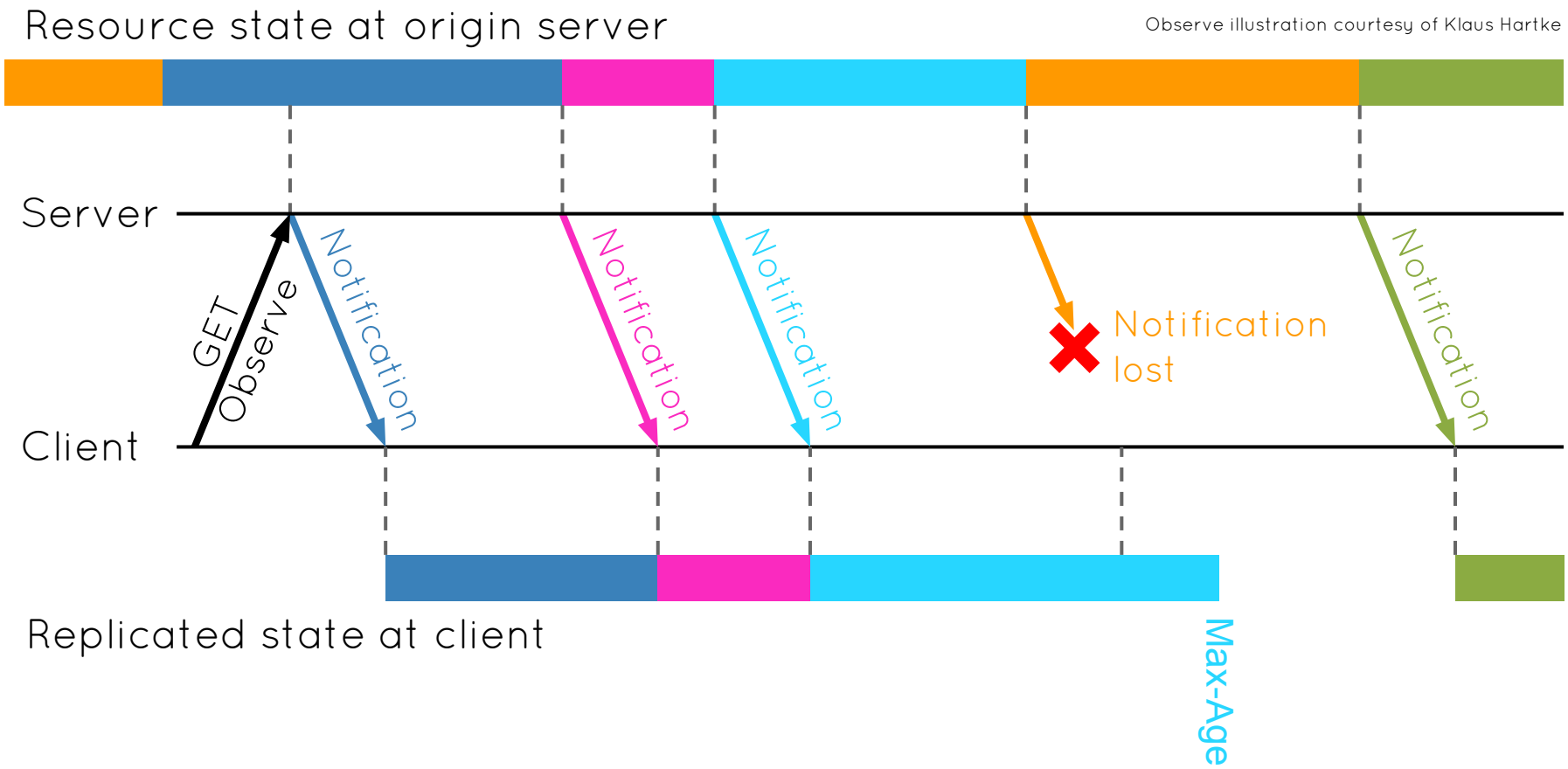
- Low parsing complexity
- Small message size

## Options

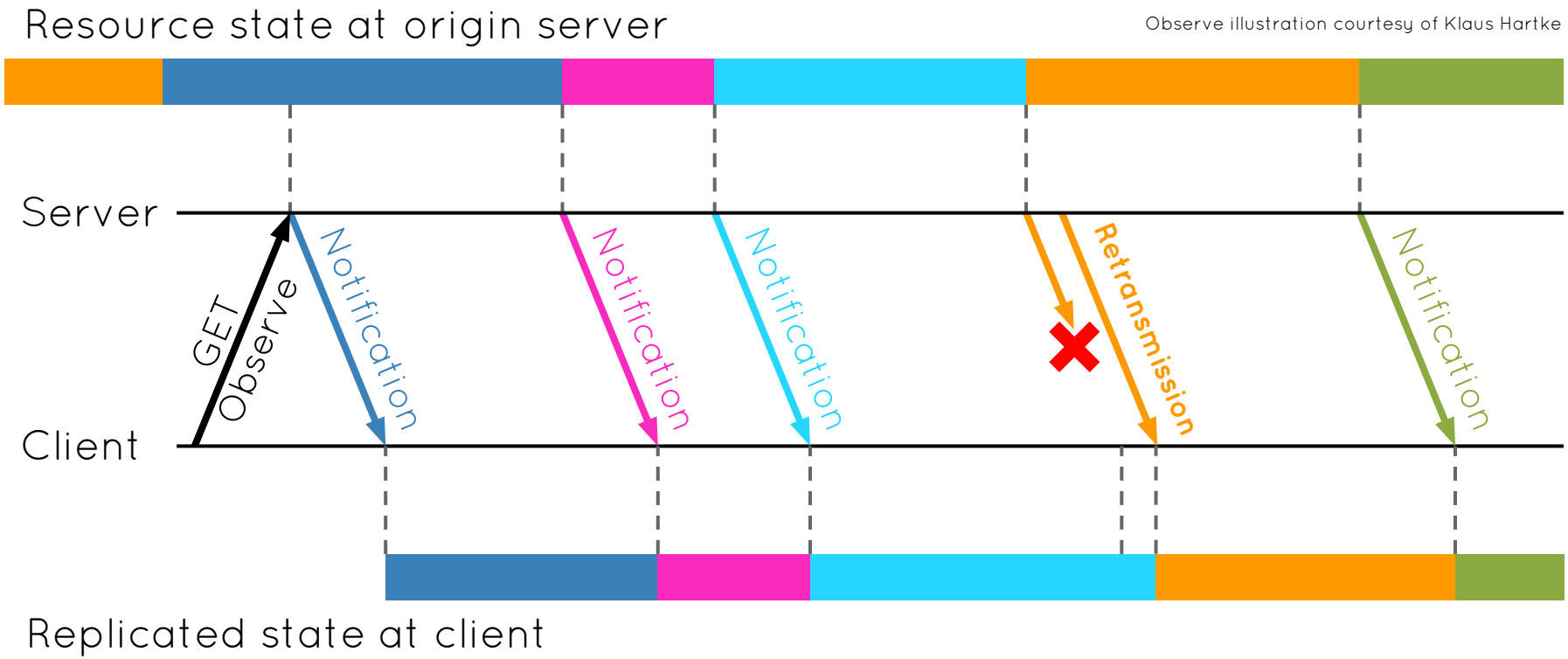
- Numbers in IANA registry
- Type-Length-Value
- Special option header marks payload if present



# Observing resources



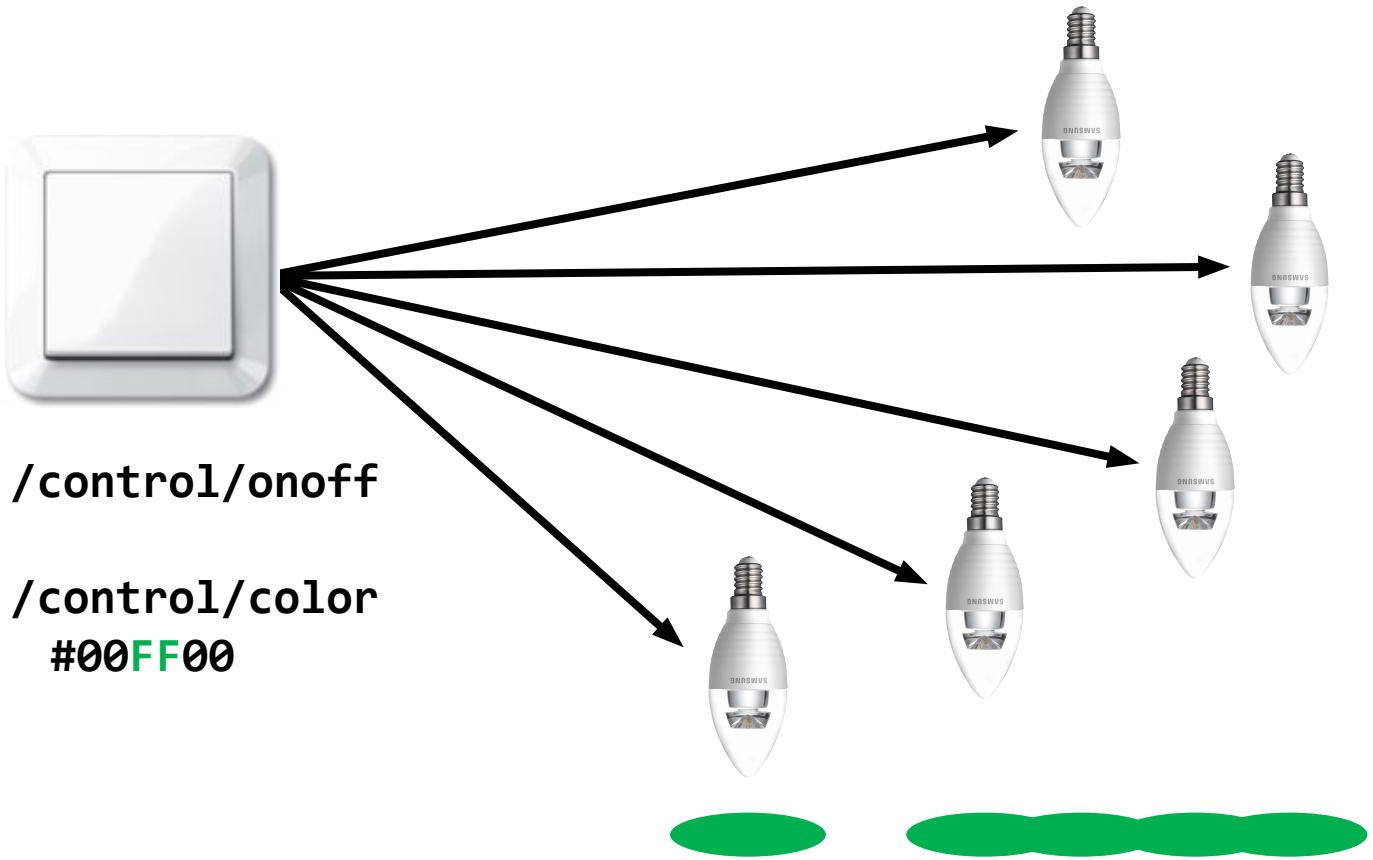
# Observing resources - CON mode



# RESTful group communication

```
GET /status/power
```

```
all-lights.floor-d.example.com
```



```
PUT /control/onoff
```

```
PUT /control/color
      #00FF00
```

# Resource discovery

Based on **Web Linking** (RFC5988)

Extended to **Core Link Format** (RFC6690)

**GET /.well-known/core**

```
</config/groups>;rt="core.gp";ct=39,
</sensors/temp>;rt="ucum.Cel";ct="0 50";obs,
</large>;rt="block";sz=1280,
</device>;title="Device management"
```

Decentralized discovery **Multicast Discovery**

Infrastructure-based **Resource Directories**

# Alternative transports

Short Message Service (SMS)


Unstructured Supplementary Service Data (USSD)

***101#** 📞

Addressable through URIs

```
coap+sms://+123456789/bananas/temp*
```

Could power up subsystems for IP connectivity after SMS signal

* illustration only, +123456789 unfortunately not allowed by URI

# Security

Based on **DTLS** (TLS/SSL for Datagrams)
Focus on Elliptic Curve Cryptography (**ECC**)
Pre-shared secrets, certificates, or raw public keys

Hardware acceleration in IoT devices

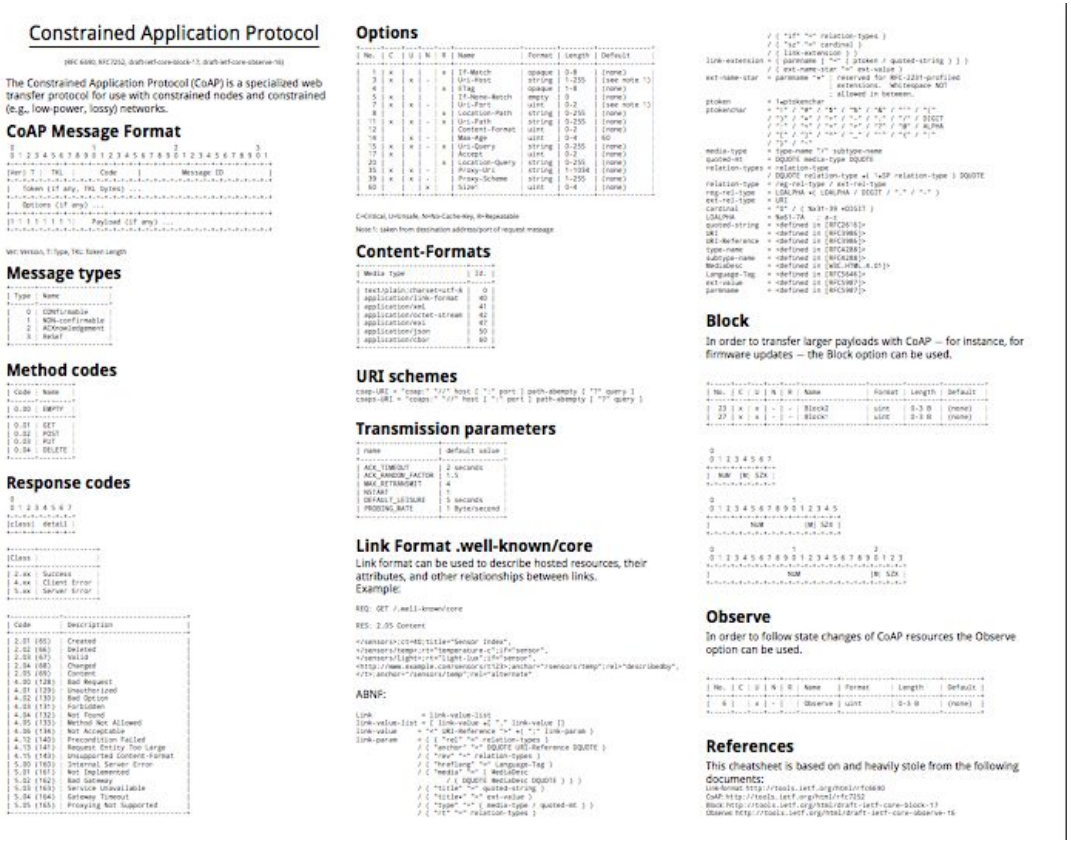
IETF is currently working on

- Authentication/authorization (ACE)
- DTLS profiles (DICE)

e.g.,

# CoAP Cheatsheet

https://github.com/markushx/coap-cheatsheet/raw/master/coap-cheatsheet.pdf



Constrained Application Protocol

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks.

CoAP Message Format

Message types

Method codes

Response codes

Options

Content-Formats

URI schemes

Transmission parameters

Link Format .well-known/core

Link format can be used to describe hosted resources, their attributes, and other relationships between links.
Example:

ABNF:

Block

In order to transfer larger payloads with CoAP – for instance, for firmware updates – the Block option can be used.

Observe

In order to follow state changes of CoAP resources the Observe option can be used.

References

This cheatsheet is based on and heavily stole from the following documents:

# Status of CoAP


Proposed Standard since 15 Jul 2013

## RFC 7252

Next working group documents in the queue

- Observing Resources
- Group Communication
- Blockwise Transfers

- Resource Directory
- HTTP Mapping Guidelines

# Status of CoAP

In use by

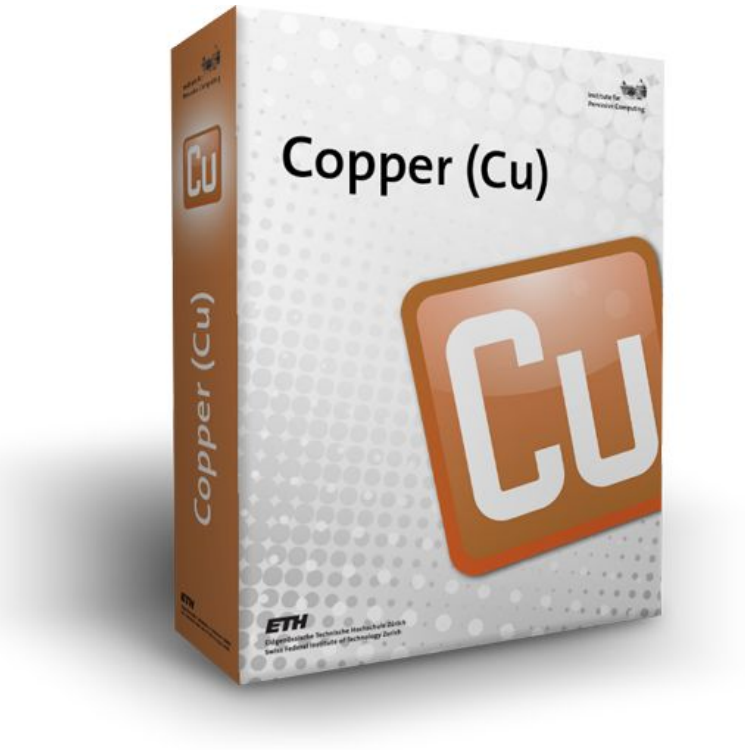

- OMA Lightweight M2M
- IPSO Alliance
- ETSI M2M / OneM2M



- Device management for network operators
- Lighting systems for smart cities

# CoAP live with Copper!

CoAP protocol handler for Mozilla Firefox

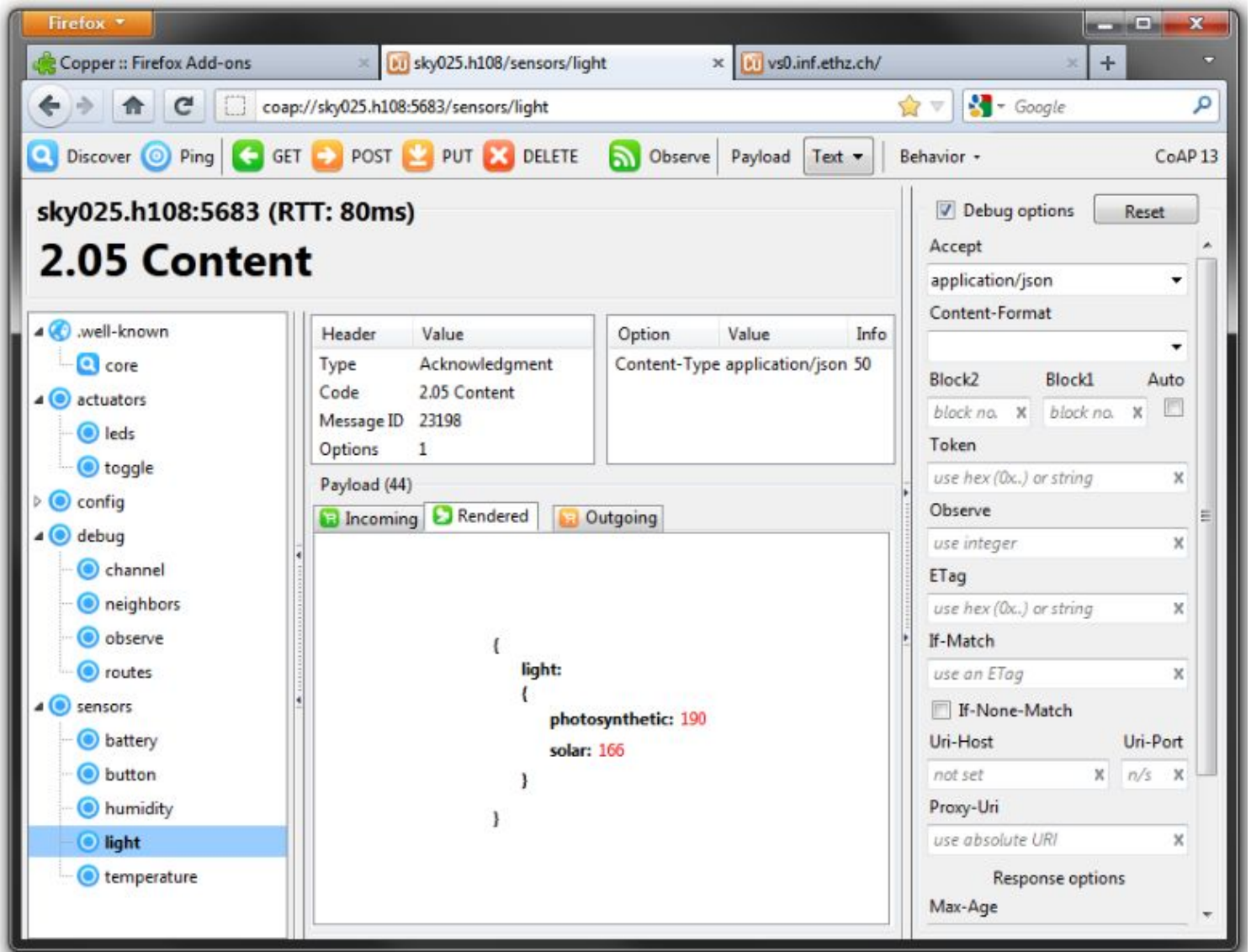



Browsing and bookmarking of CoAP URIs

Interaction with resource like RESTClient or Poster

Treat tiny devices like normal RESTful Web services

# Copper (Cu) CoAP user-agent

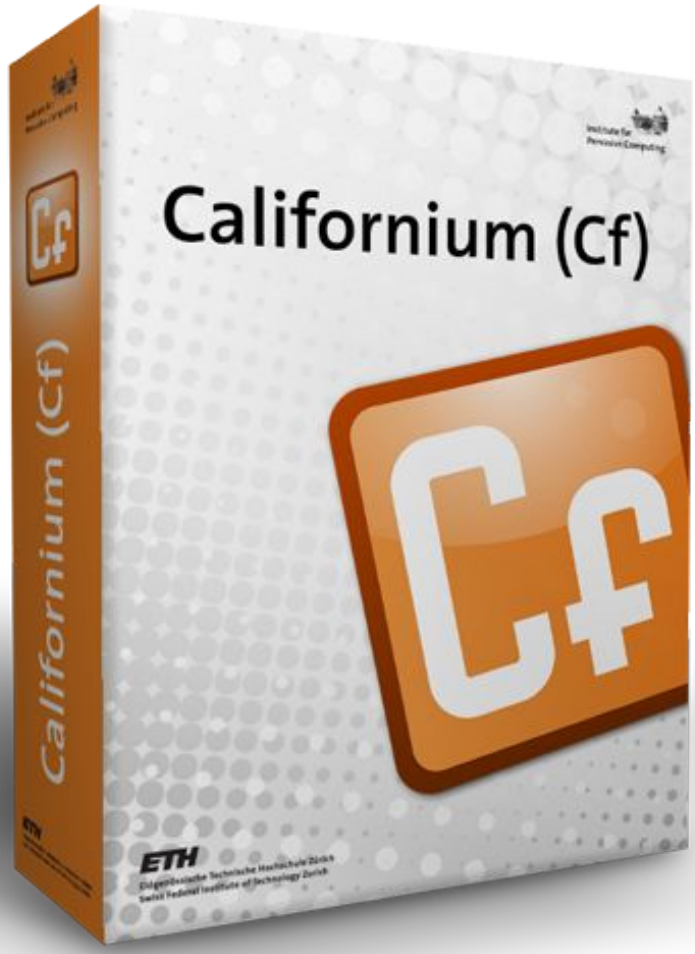


# CoAP live with Copper!

Available sandboxes:

**coap://californium.eclipse.org:5683/**

**coap://coap.me:5683/**


Californium (Cf)
Californium (Cf)
ETH

# Californium (Cf) CoAP framework

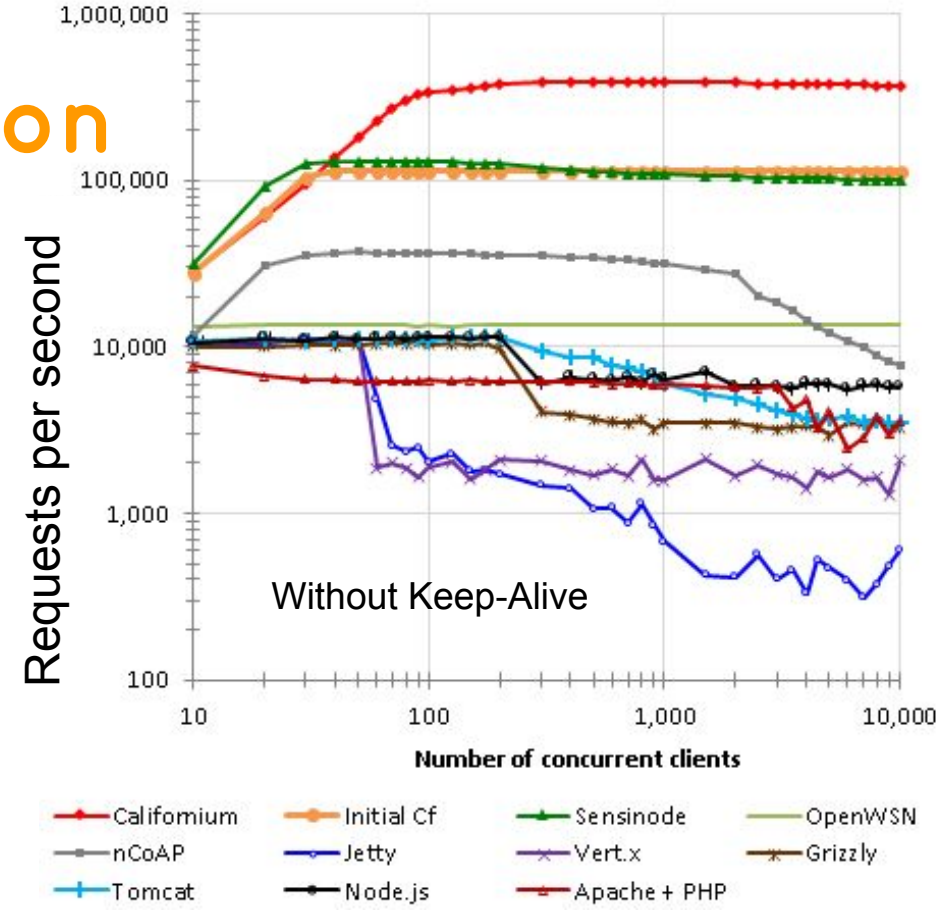
Unconstrained CoAP implementation

- written in Java
- focus on scalability and usability

For

- IoT cloud services
- Stronger IoT devices
  (Java SE Embedded or special JVMs)

# Paper on evaluation at IoT 2014




Matthias Kovatsch, Martin Lanter, and Zach Shelby.
*Scalable Cloud Services for the Internet of Things.*
In Proc. IoT, Cambridge, MA, USA, 2014.

http://www.iot-conference.org/iot2014/

# Let’s get concrete!

# Project structure

## 6 repositories on GitHub

- https://github.com/eclipse/californium
  Parent POM
- https://github.com/eclipse/californium.core
  Core library and example projects
- https://github.com/eclipse/californium.element-connector
  Abstraction for modular network stage (Connectors)
- https://github.com/eclipse/californium.scandium
  DTLS 1.2 implementation for network stage (DtlsConnector)
- https://github.com/eclipse/californium.tools
  Stand-alone CoAP tools such as console client or RD
- https://github.com/eclipse/californium.actinium
  App server for server-side JavaScript*

*not yet ported to new implementation and using deprecated CoAP draft version

# Maven artifacts

Maven artifacts available at

https://repo.eclipse.org/content/repositories/californium-snapshots/

https://repo.eclipse.org/content/repositories/californium-releases/

And on Maven Central too

# Code structure

https://github.com/eclipse/californium.core

- Libraries ("californium-" prefix)
  - **californium-core** CoAP, client, server
  - **californium-osgi** OSGi wrapper
  - **californium-proxy** HTTP cross-proxy

- Example code
- Example projects ("cf-" prefix)

# Code structure

https://github.com/eclipse/californium.core

- Libraries
- Example code
  - **cf-api-demo** API call snippets
- Example projects

# Code structure

https://github.com/eclipse/californium.core

- Libraries
- Example code
- Example projects
  - **cf-helloworld-client** basic GET client
  - **cf-helloworld-server** basic server
  - **cf-plugtest-checker** tests Plugtest servers
  - **cf-plugtest-client** tests client functionality
  - **cf-plugtest-server** tests server functionality
  - **cf-benchmark** performance tests
  - **cf-secure** imports Scandium (DTLS)
  - **cf-proxy** imports californium-proxy

# Server API

Important classes (see org.eclipse.californium.core)

- **CoapServer**
- **CoapResource**
- **CoapExchange**

- Implement custom resources
  by extending **CoapResource**
- Add resources to server
- Start server

# Server API - resources

```
import static org.eclipse.californium.core.coap.CoAP.ResponseCode.*; // shortcuts

public class MyResource extends CoapResource {
    @Override
    public void handleGET(CoapExchange exchange) {
        exchange.respond("hello world"); // reply with 2.05 payload (text/plain)
    }
    @Override
    public void handlePOST(CoapExchange exchange) {
        exchange.accept(); // make it a separate response

        if (exchange.getRequestOptions()....) {
            // do something specific to the request options
        }
        exchange.respond(CREATED); // reply with response code only (shortcut)
    }
}
```

# Server API - creation

```
public static void main(String[] args) {

    CoapServer server = new CoapServer();

    server.add(new MyResource("hello"));

    server.start(); // does all the magic
}
```

# Client API

Important classes

- **CoapClient**
- **CoapHandler**
- **CoapResponse**
- **CoapObserveRelation**

- Instantiate **CoapClient** with target URI
- Use offered methods get(), put(), post(), delete(), observe(), validate(), discover(), or ping()
- Optionally define **CoapHandler** for asynchronous requests and observe

# Client API - synchronous

```
public static void main(String[] args) {

        CoapClient client1 = new CoapClient("coap://californium..eclipse.org:5683/multi-format");

        String text = client1.get().getResponseText(); // blocking call
        String xml = client1.get(APPLICATION_XML).getResponseText();

        CoapClient client2 = new CoapClient("coap://californium.eclipse.org:5683/test");

        CoapResponse resp = client2.put("payload", TEXT_PLAIN); // for response details
        System.out.println( resp.isSuccess() );
        System.out.println( resp.getOptions() );

        client2.useNONs();  // use autocomplete to see more methods
        client2.delete();
        client2.useCONs().useEarlyNegotiation(32).get(); // it is a fluent API
}
```

# Client API - asynchronous

```
public static void main(String[] args) {

    CoapClient client = new CoapClient("coap://californium.eclipse.org:5683/separate");

    client.get(new CoapHandler() { // e.g., anonymous inner class

        @Override public void onLoad(CoapResponse response) { // also error resp.
            System.out.println( response.getResponseText() );
        }

        @Override public void onError() { // I/O errors and timeouts
            System.err.println("Failed");
        }
    });
}
```

# Client API - observe

```
public static void main(String[] args) {

    CoapClient client = new CoapClient("coap://californium.eclipse.org:5683/obs");

    CoapObserveRelation relation = client.observe(new CoapHandler() {

        @Override public void onLoad(CoapResponse response) {
            System.out.println( response.getResponseText() );
        }

        @Override public void onError() {
            System.err.println("Failed");
        }
    });

    relation.proactiveCancel();
}
```

# Advanced API

Get access to internal objects with

advanced() on
**CoapClient**, **CoapResponse**, **CoapExchange**

Define your own concurrency models with

**ConcurrentCoapResource** and
CoapClient.useExecutor() / setExecutor(exe)

# HANDS-ON!

# Getting started

- Tutorial projects
  https://github.com/jvermillard/hands-on-coap

- Launch Eclipse
- Import projects contained on the USB stick
  - File > Import... > Existing projects into workspace

# Step 1

**The mandatory Hello world CoAP server!**

1. Complete the code:
   Add “hello” resource with a custom message (GET)
   Run the CoAP server

2. Test with Copper

# Step 2

**Improve the server by adding:**

1. A “subpath/another” hello world
2. Current time in milliseconds
3. A writable resource
4. A removable resource

# Step 3

## Hello world CoAP client

1. Complete the code for reading the previous “helloworld” values

2. Connect your client with your server

# More fun

Connect with the LED strip

Read the sensors

Change the color

Have fun!

# Where is the code?

## Tutorial steps

https://github.com/jvermillard/hands-on-coap

## Californium

https://github.com/eclipse?query=californium

# Hands-off

Questions?

# Going further with CoAP

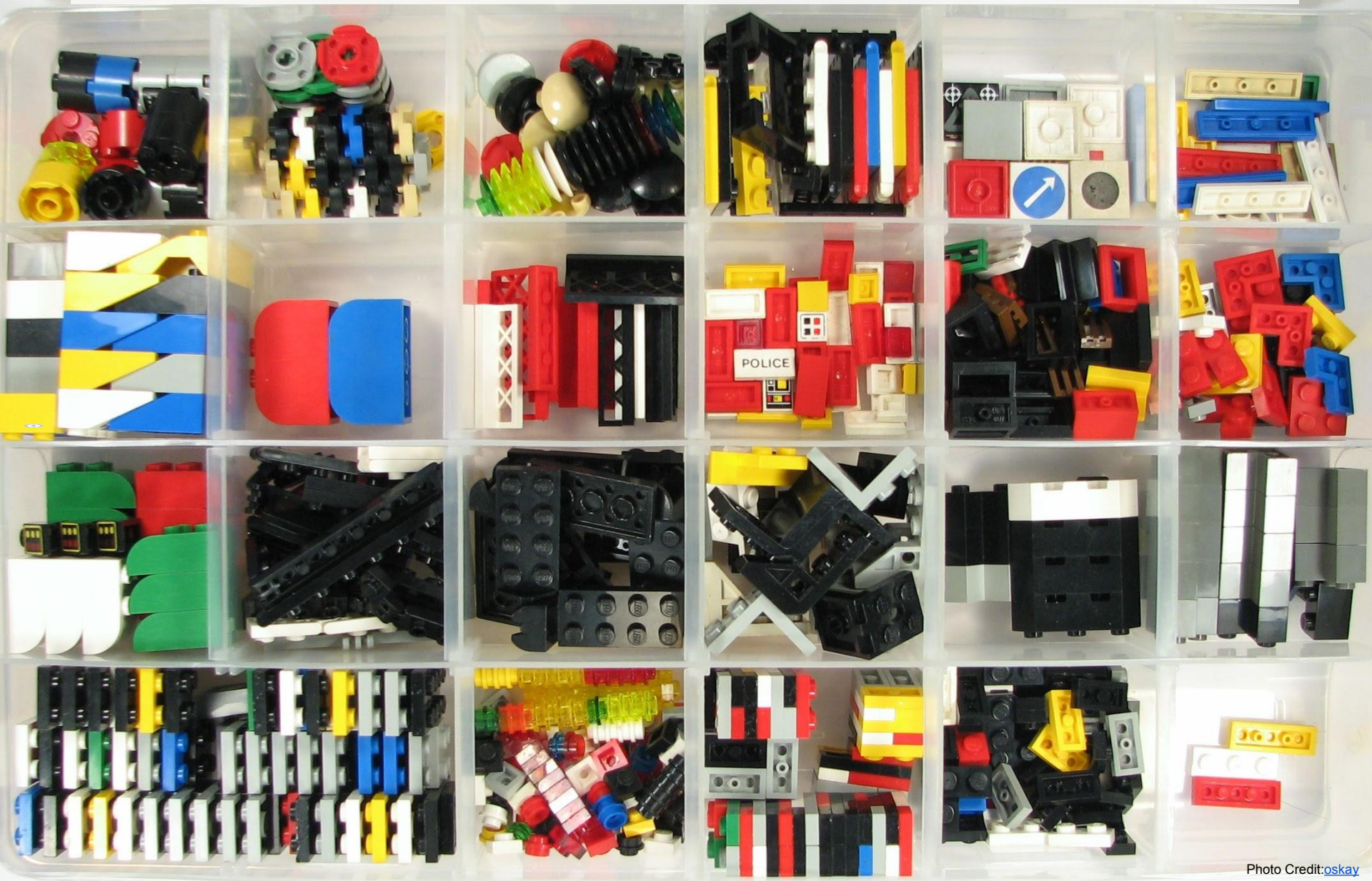



Photo Credit:oskay

# Going further with CoAP

**Scandium (Sc)**

DTLS (TLS/SSL for UDP) for adding security

**Californium (Cf) Proxy**

HTTP/CoAP proxy

**Californium (Cf) RD**

CoAP resource directory

# Going further

**Contiki OS**

Connects tiny, low-power MCU to the Internet

http://contiki-os.org

**Microcoap**

CoAP for arduino

https://github.com/1248/microcoap

# OMA Lightweight M2M

An device management protocol

Created by the Open Mobile Alliance

Configure, monitor, upgrade your device using CoAP over UDP and SMS

In a RESTful way!

# OMA Lightweight M2M

**The specification**

http://technical.openmobilealliance.org

**C client library**

http://github.com/eclipse/wakaama

**Java server implementation**

http://github.com/eclipse/leshan

# Thanks!

More questions? Feel free to contact us!

**Benjamin Cabé**

**@kartben / benjamin@eclipse.org**

**Matthias Kovatsch**

kovatsch@inf.ethz.ch

**Julien Vermillard**

@vrmvrm / jvermillard@sierrawireless.com