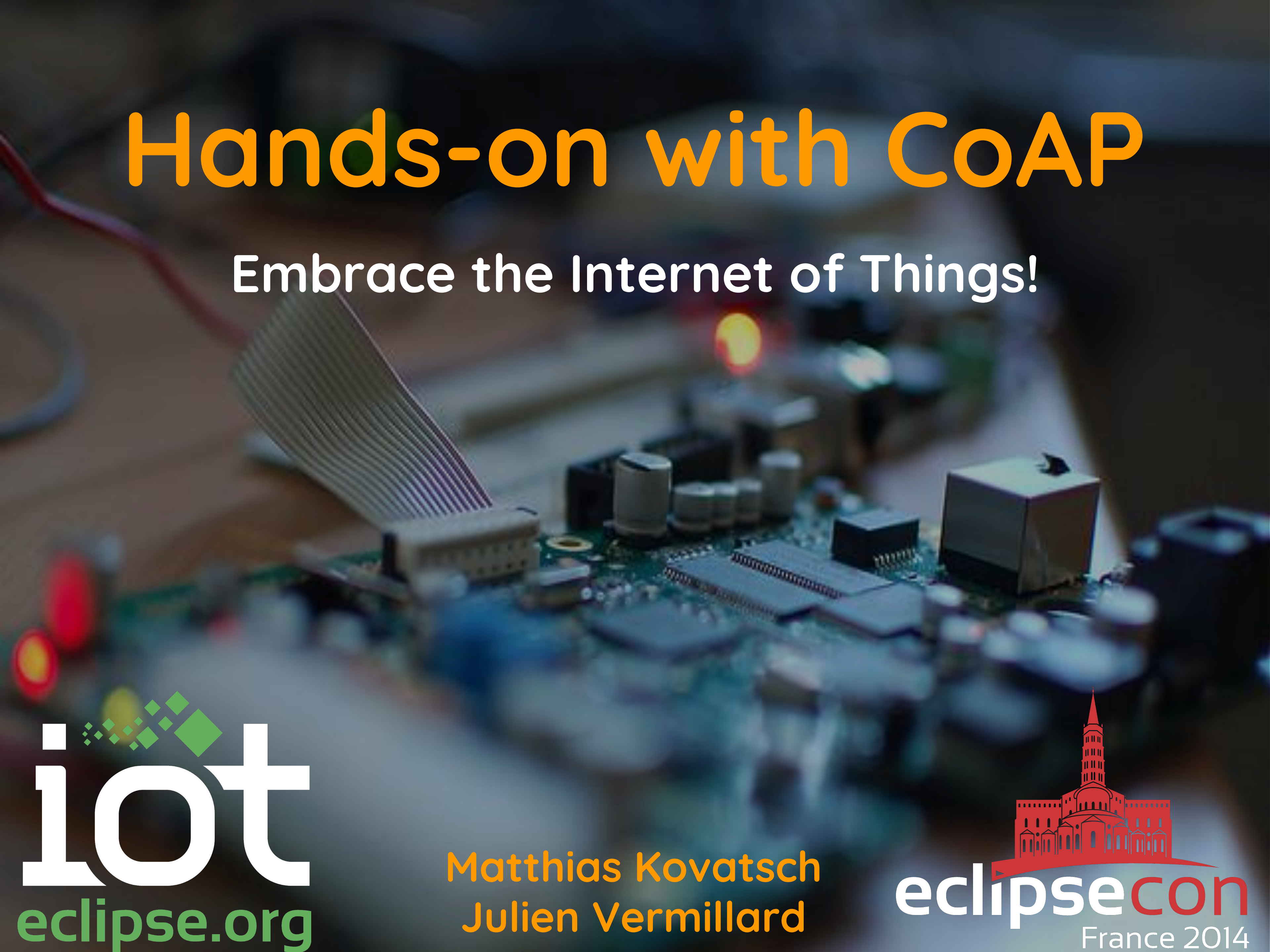


Hands-on with CoAP

Embrace the Internet of Things!

iot
eclipse.org

Matthias Kovatsch
Julien Vermillard

eclipsecon
France 2014

Follow the slides

http://goo.gl/LLQ03w

Your devoted presenters :-)

Julien Vermillard / @vrmvrm

Software Engineer at Sierra Wireless
http://airvantage.net M2M Cloud

Apache member, Eclipse committer on Californium and Wakaama

More IoT stuff:
https://github.com/jvermillard

Your devoted presenters :-)

Matthias Kovatsch

Researcher at ETH Zurich, Switzerland
Focus on Web technology for the IoT

IETF contributor in CoRE and LWIG

Author of Californium (Cf),
Erbium (Er), and Copper (Cu)

http://people.inf.ethz.ch/mkovatsc

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

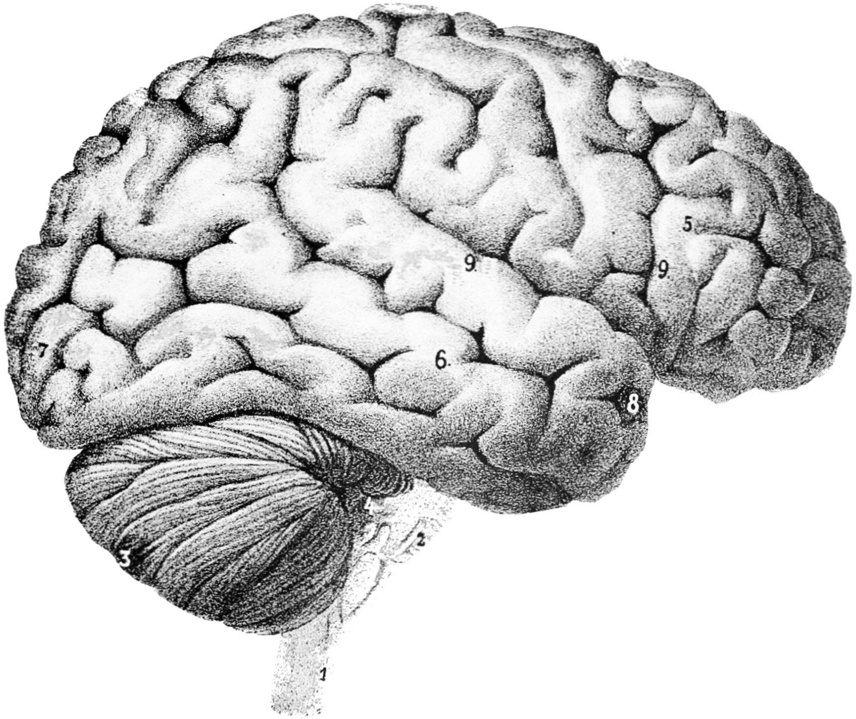
Eclipse IDE

Basic Java knowledge

Californium JARs

Firefox + Copper

Your brainzzz



Content 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 statististics, 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

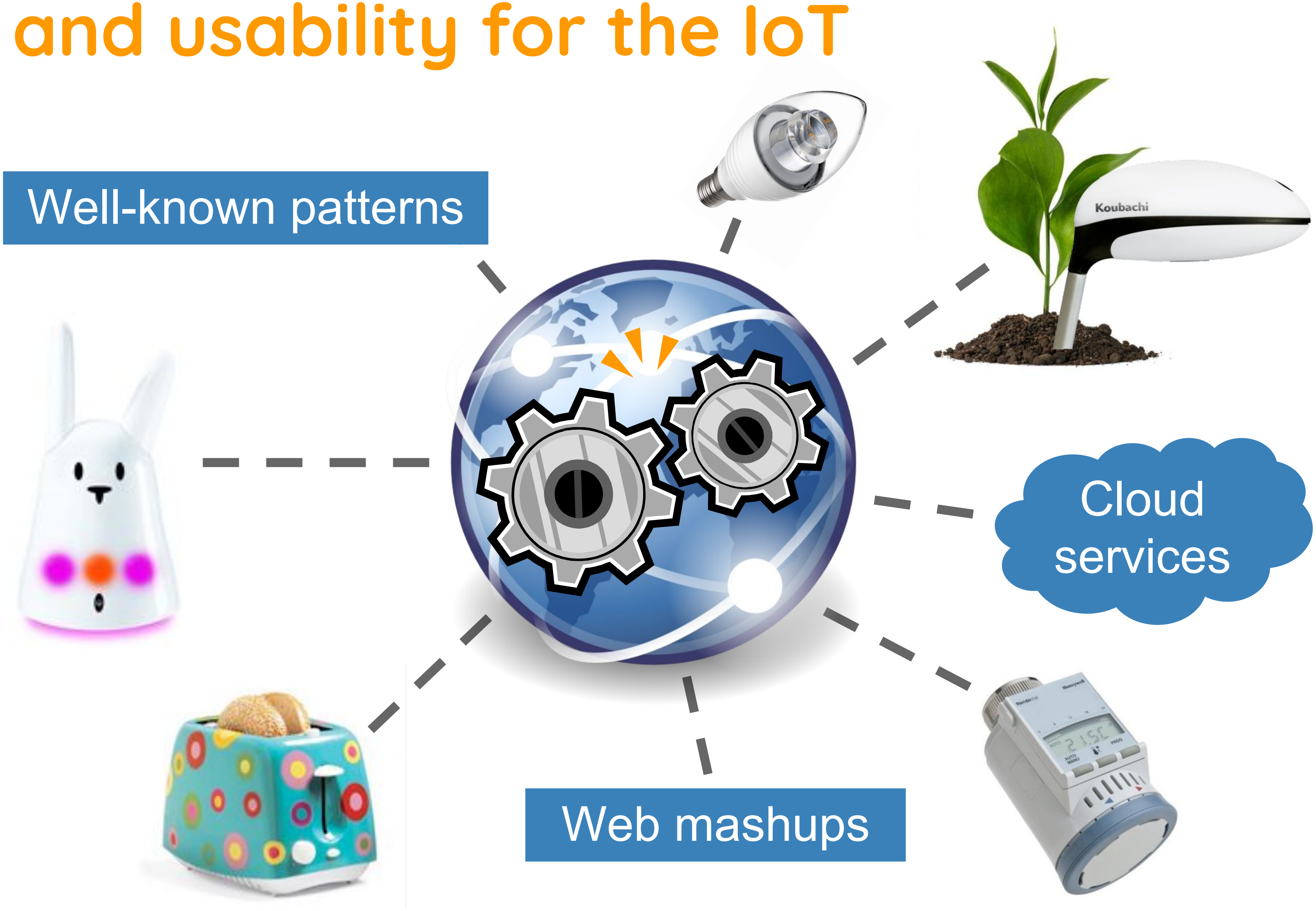
Display sensor data on mobile app, dashboards,

HTTP, Websockets, …

The Web of Things

Slide courtesy
of Vlad Trifa

Application-layer interoperability and usability for the IoT



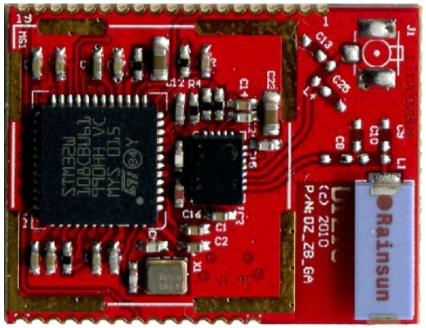
Well-known patterns

Cloud services

Web mashups

Tiny Resource-constrained devices

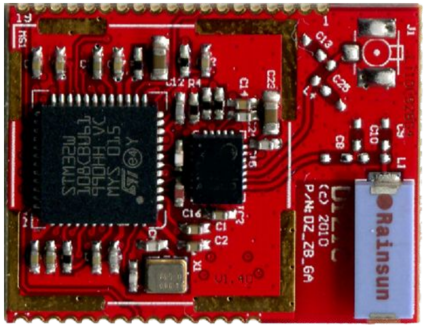
Class 1 devices
~100KiB Flash
~10KiB RAM



Low-power networks

Tiny Resource-constrained devices

Target
of less than $1
for IoT SoC



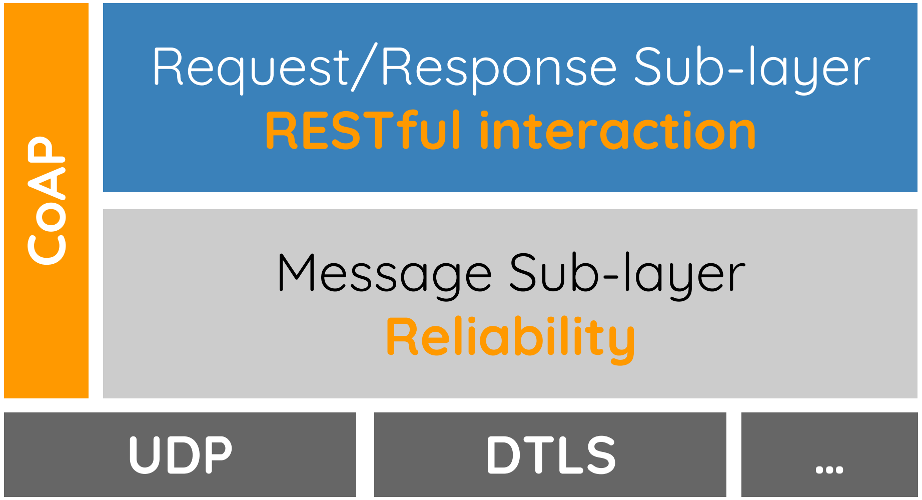
TCP and HTTP
are not a good fit

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”)

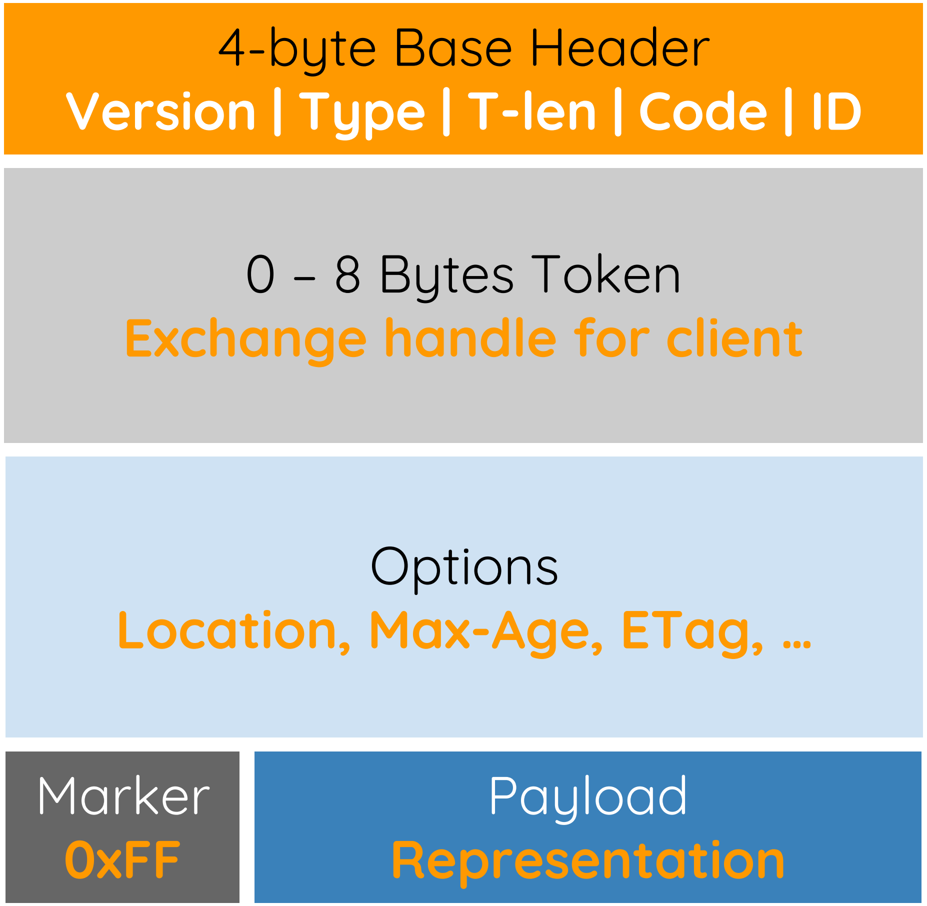
Constrained Application Protocol

Binary protocol

- Low parsing complexity
- Small message size

Options

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



4-byte Base Header

Version | Type | T-len | Code | ID

0 – 8 Bytes Token

Exchange handle for client

Options

Location, Max-Age, ETag, …

Marker

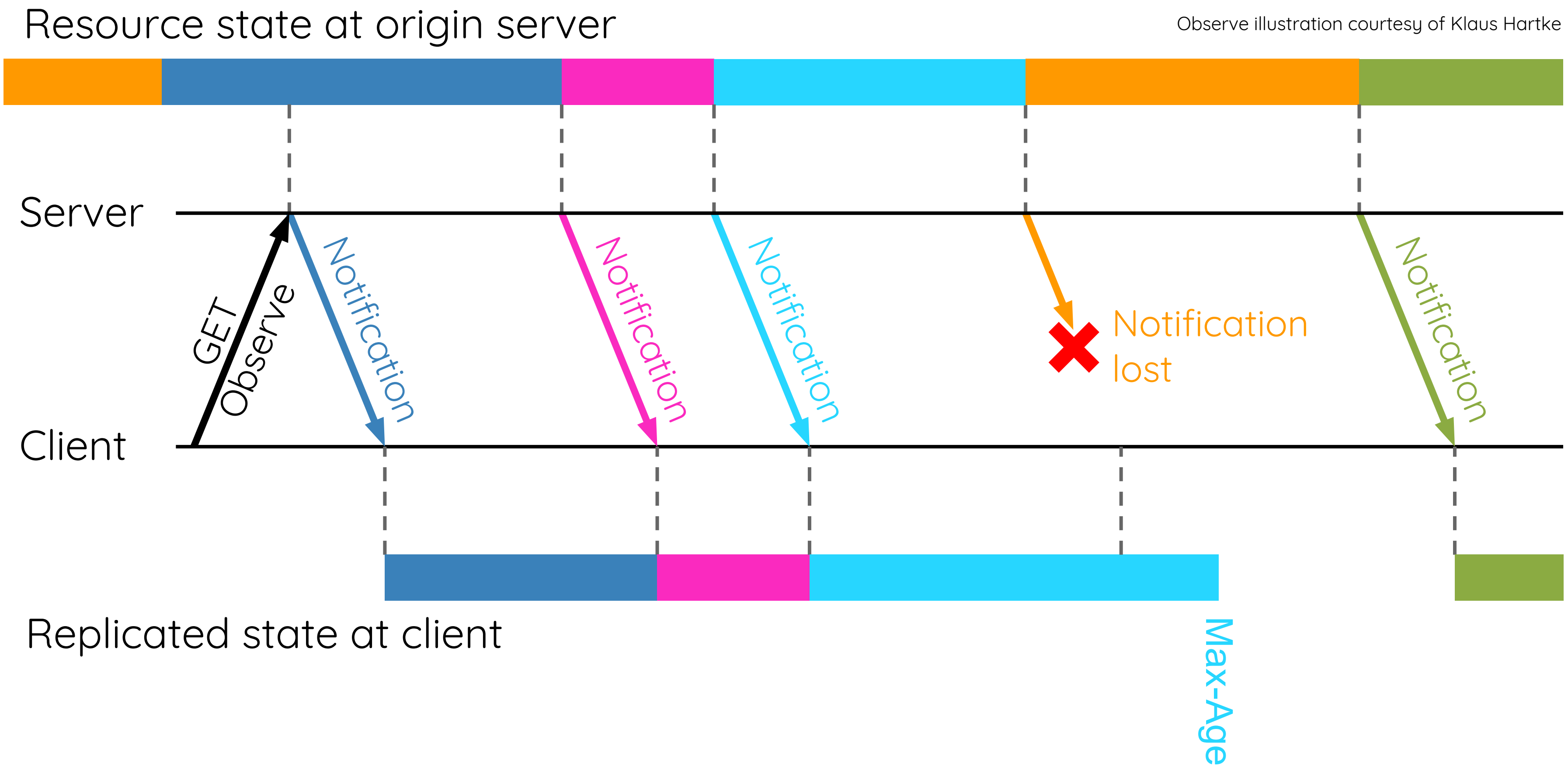
0xFF

Payload

Representation

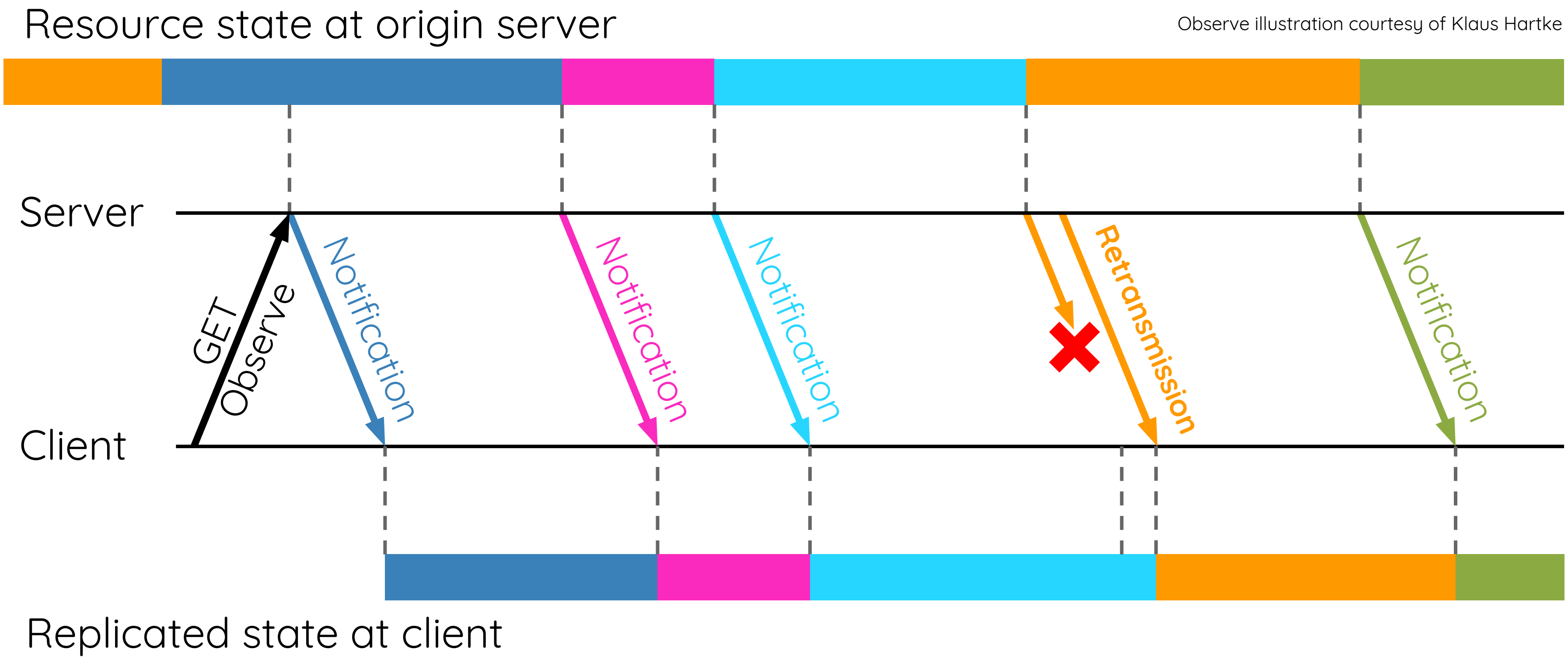
Observing resources

Observe illustration courtesy of Klaus Hartke



Observing resources - CON mode

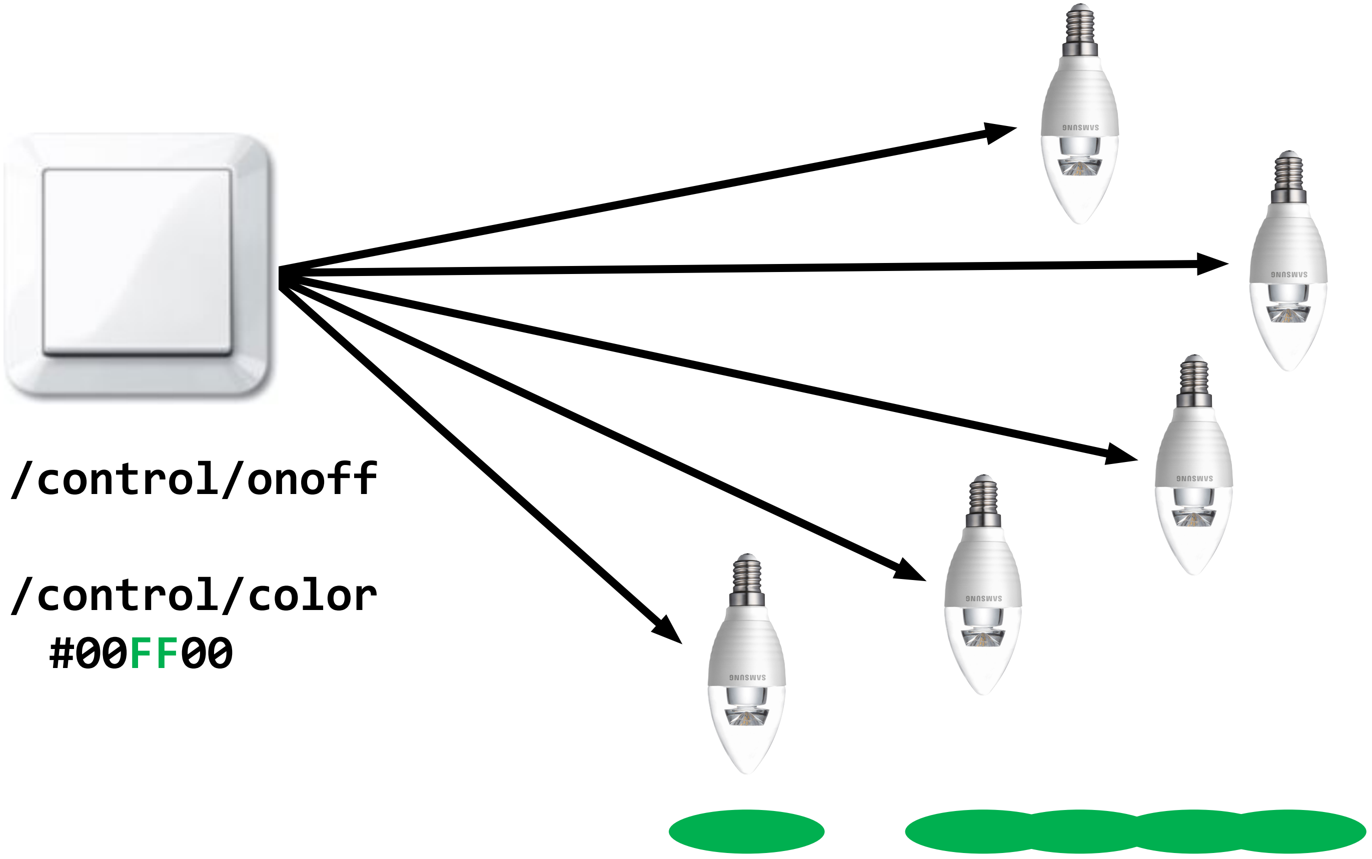
Observe illustration courtesy of Klaus Hartke



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

e.g.,

IETF is currently working on

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

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

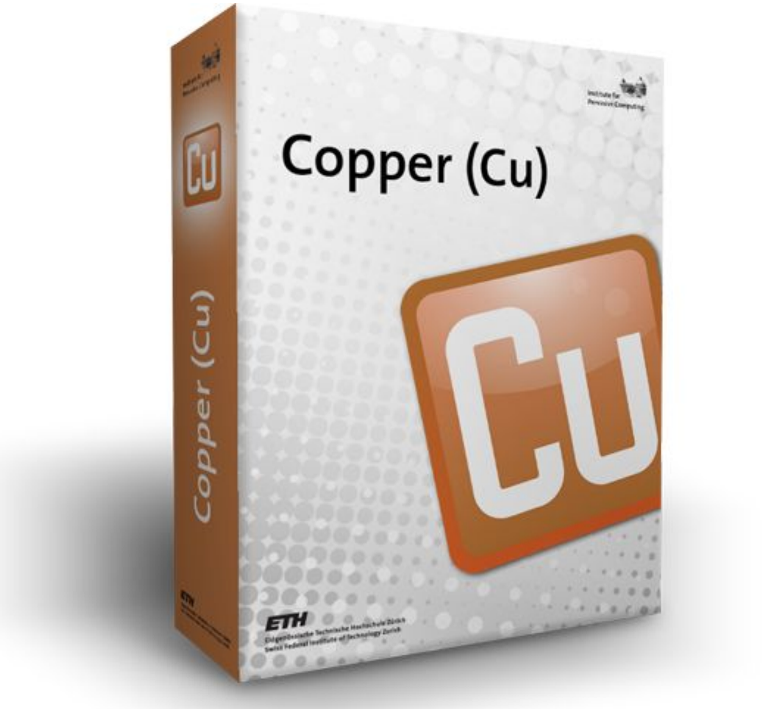
In use by

- 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



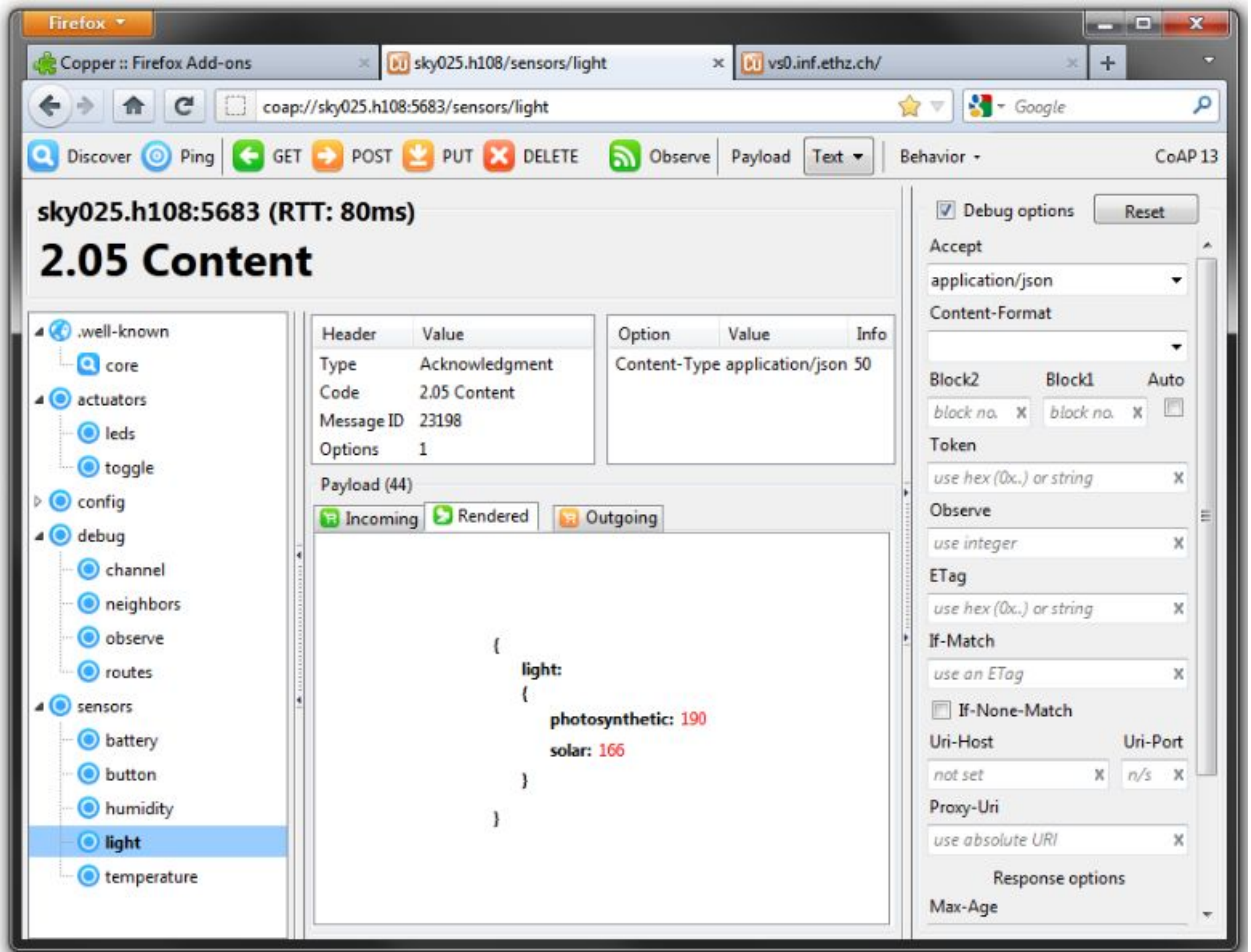
Browsing and bookmarking of CoAP URIs

Interaction with resource like RESTClient or Poster

Treat tiny devices like normal RESTful Web services

https://github.com/mkovatsc/Copper
https://addons.mozilla.org/en-US/firefox/addon/copper-270430/

Copper (Cu) CoAP user-agent



coap://sky025.h108:5683/sensors/light

sky025.h108:5683 (RTT: 80ms)

2.05 Content

Header	Value
Type	Acknowledgment
Code	2.05 Content
Message ID	23198
Options	1

Option	Value	Info
Content-Type	application/json	50

Payload (44)

```
{
    light:
    {
        photosynthetic: 190
        solar: 166
    }
}
```

☑ Debug options

Accept: application/json

Content-Format:

Block2 | Block1 | Auto □

Token

Observe

ETag

If-Match

□ If-None-Match

Uri-Host | Uri-Port

Proxy-Uri

CoAP live with Copper!

Available sandboxes:

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

same as

coap://vs0.inf.ethz.ch:5683/

coap://coap.me:5683/

Californium (Cf)
Institute for Pervasive Computing
Cf
Californium (Cf)
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Californium (Cf) CoAP framework

Unconstrained CoAP implementation

- written in Java
- focus on scalability and usability

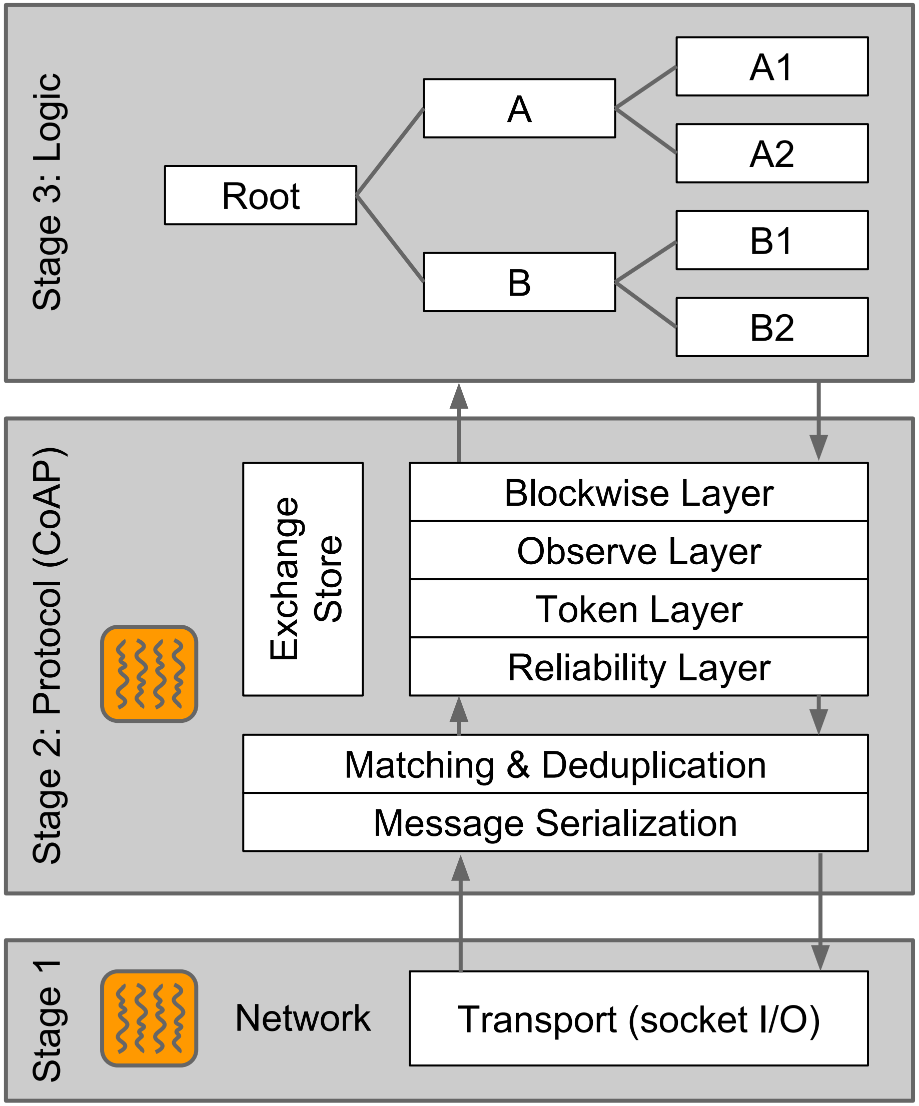
For

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

3-stage architecture

Stages

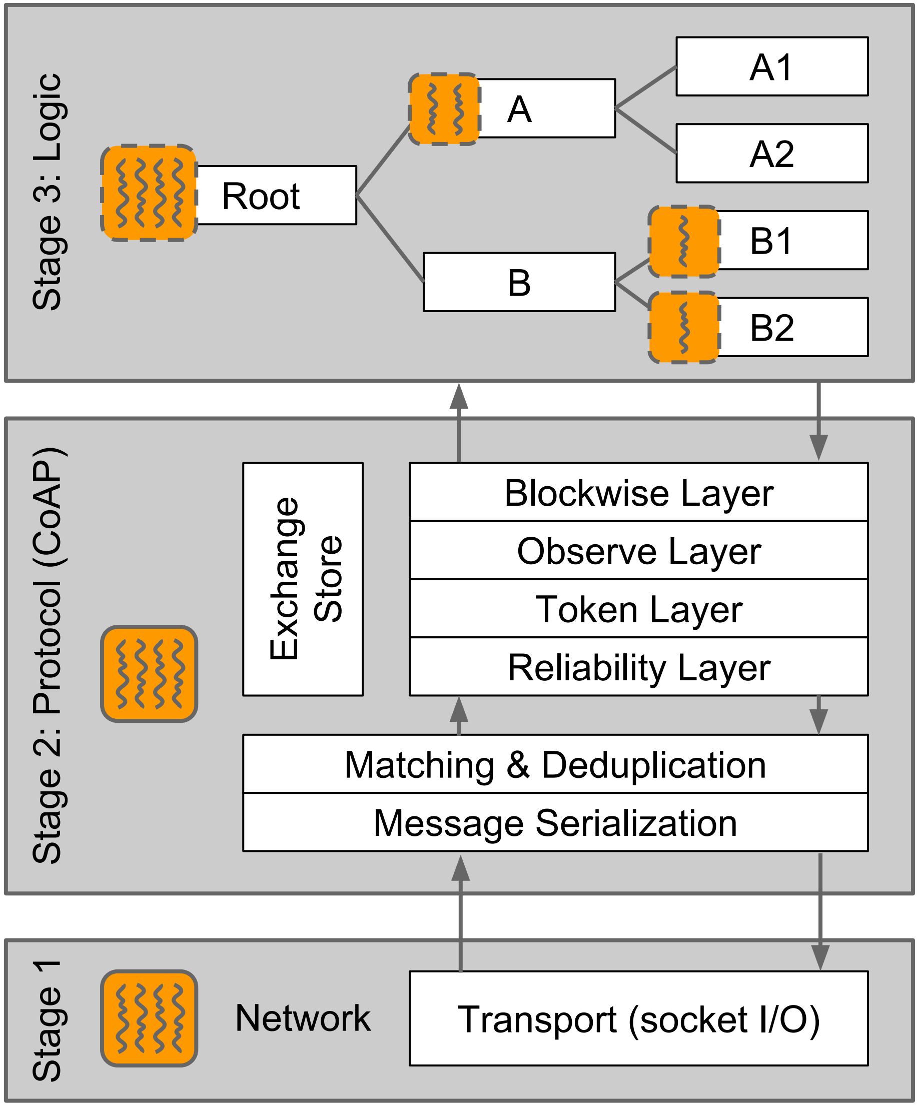
- Decoupled with message queues
- independent concurrency models
- Adjusted statically for platform/application
- Stage 1 depends on OS and transport
- Stage 2 usually one thread per core



Stage 3: server role

Web resources

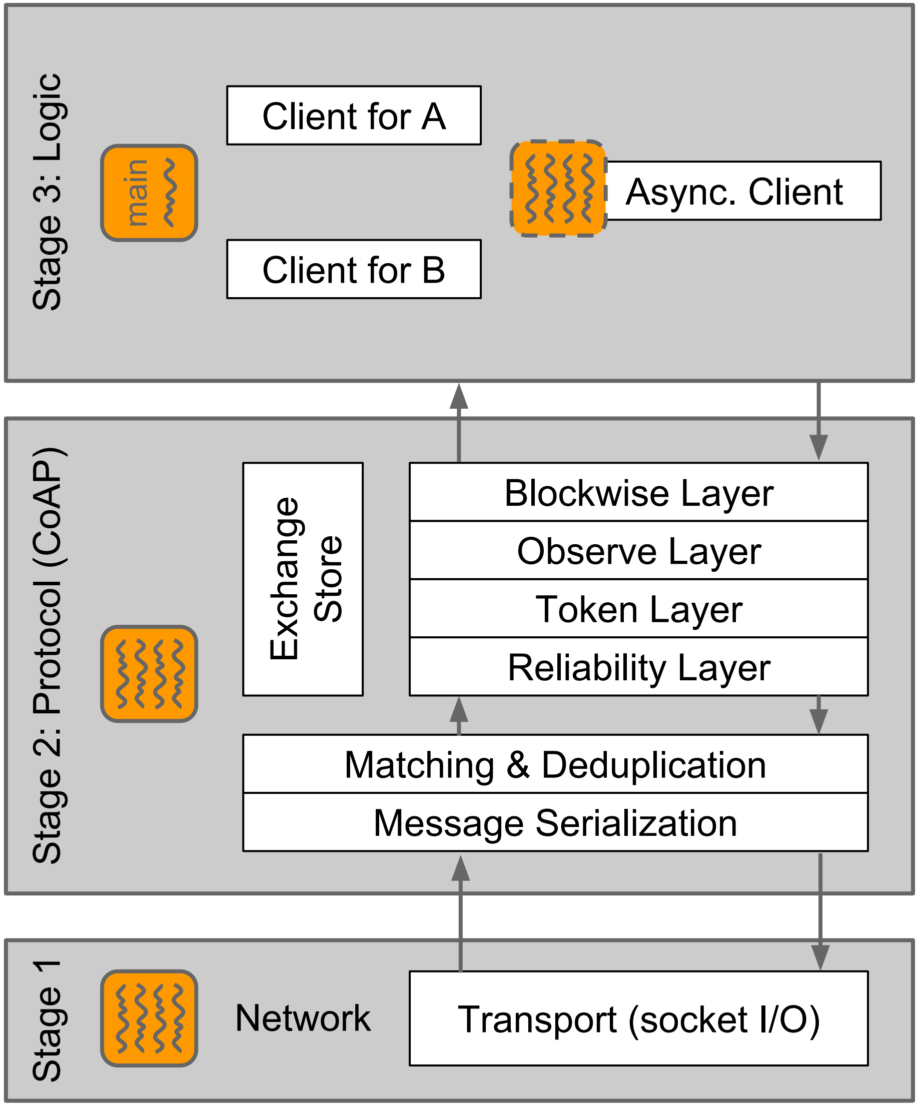
- Optional thread pool for each Web resource
- Inherited by parent or transitive ancestor
- Protocol threads used if none defined



Stage 3: client role

Clients with response handlers

- Object API called from main or user thread
- Synchronous: Protocol threads unblock API calls
- Asynchronous: Optional thread pools for response handling (e.g., when observing)



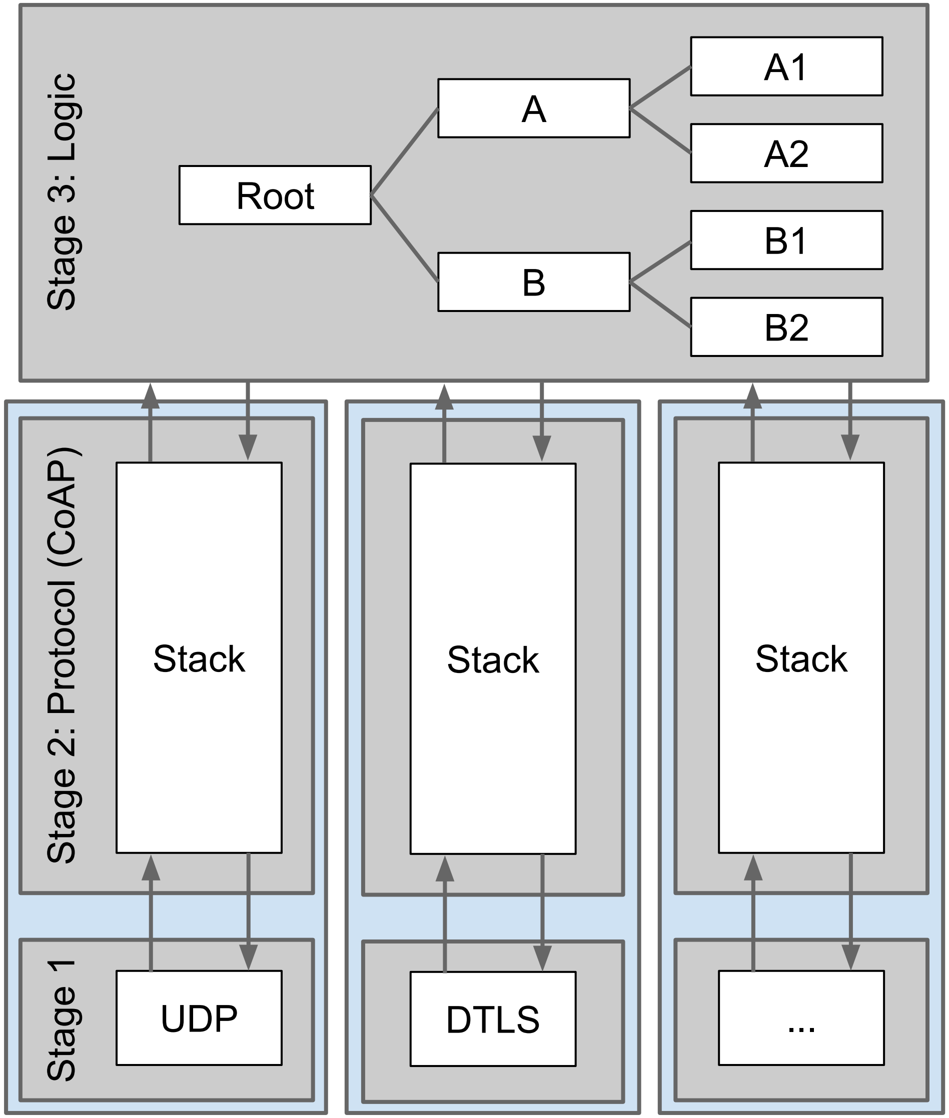
Endpoints

Encapsulate stages 1+2

Enable

- multiple channels
- stack variations for different transports

Individual concurrency models, e.g., for DTLS

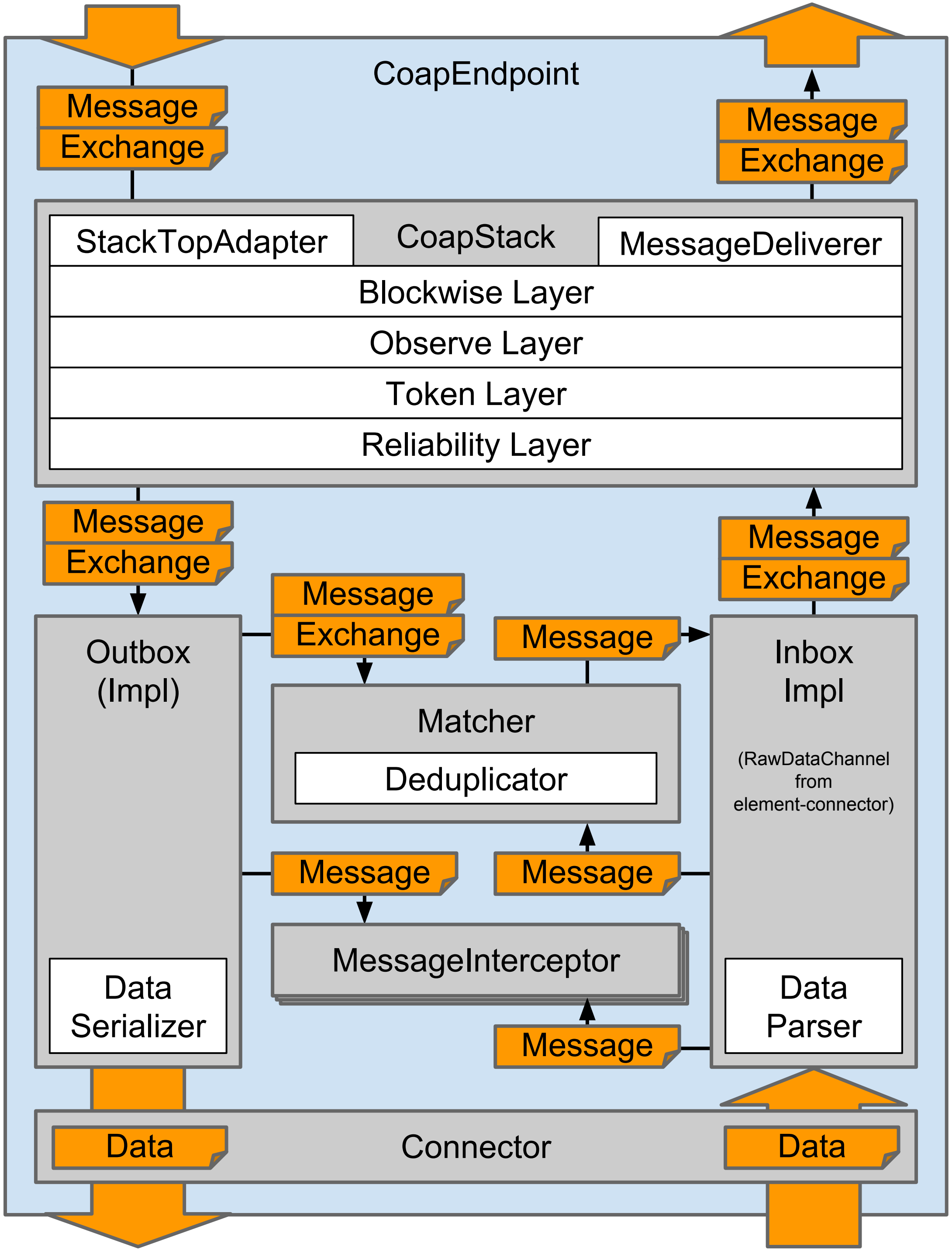


Endpoints

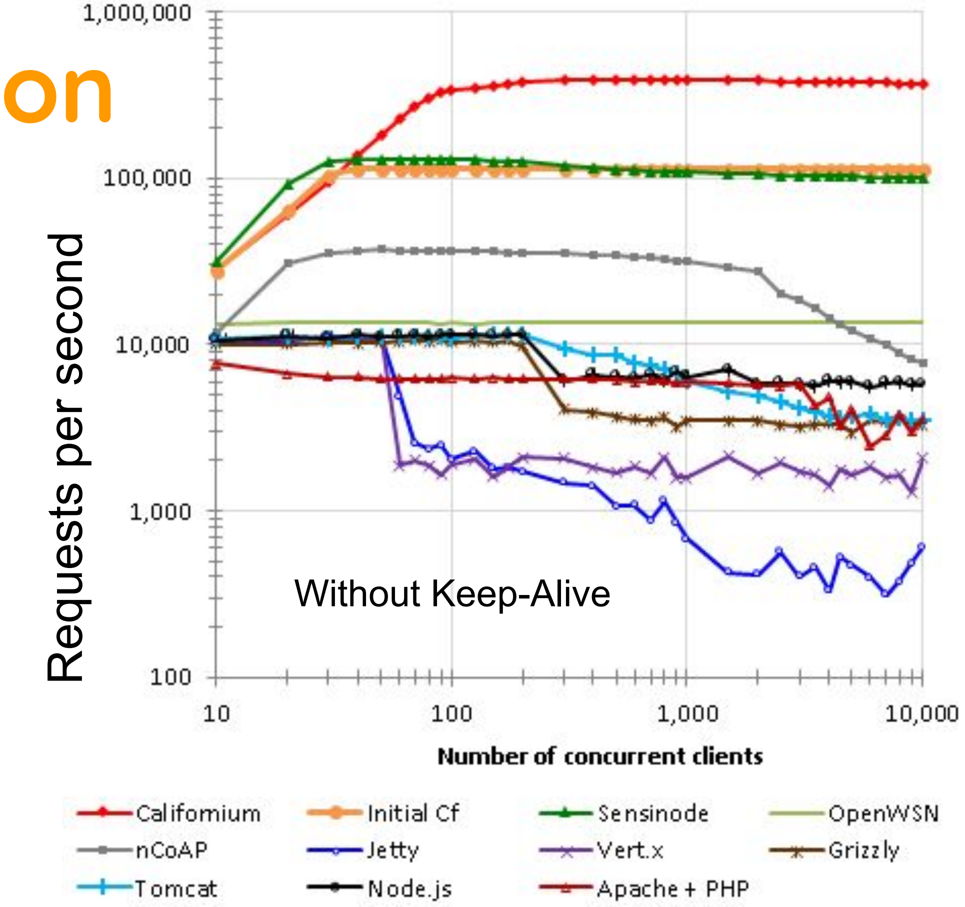
Implemented in **CoapEndpoint**

Separation of bookkeeping and processing

Exchanges carry state



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.vs.inf.ethz.ch/publ/papers/mkovatsc-2014-iot-californium.pdf

Let’s get concrete!

Project structure

Six repositories on GitHub

- https://github.com/eclipse/californium
  Parent Maven POM with project metadata
- https://github.com/eclipse/californium.element-connector
  Abstraction for datagram-based transports (UDP, DTLS)
- https://github.com/eclipse/californium.scandium
  DTLS 1.2 implementation for network stage (DtlsConnector)
- https://github.com/eclipse/californium.core
  Core libraries and example projects including Plugtest client/server
- 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 IoT mashups with JavaScript

Maven

Maven artifacts are available at

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

and releases at Maven Central

http://search.maven.org/#search|ga|1|californium

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-android** Android Studio project
  - **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://iot.eclipse.org:5683/multi-format");

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

    CoapClient client2 = new CoapClient("coap://iot.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://iot.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://iot.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**

Use clients in resource handlers with

createClient(uri);

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
   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

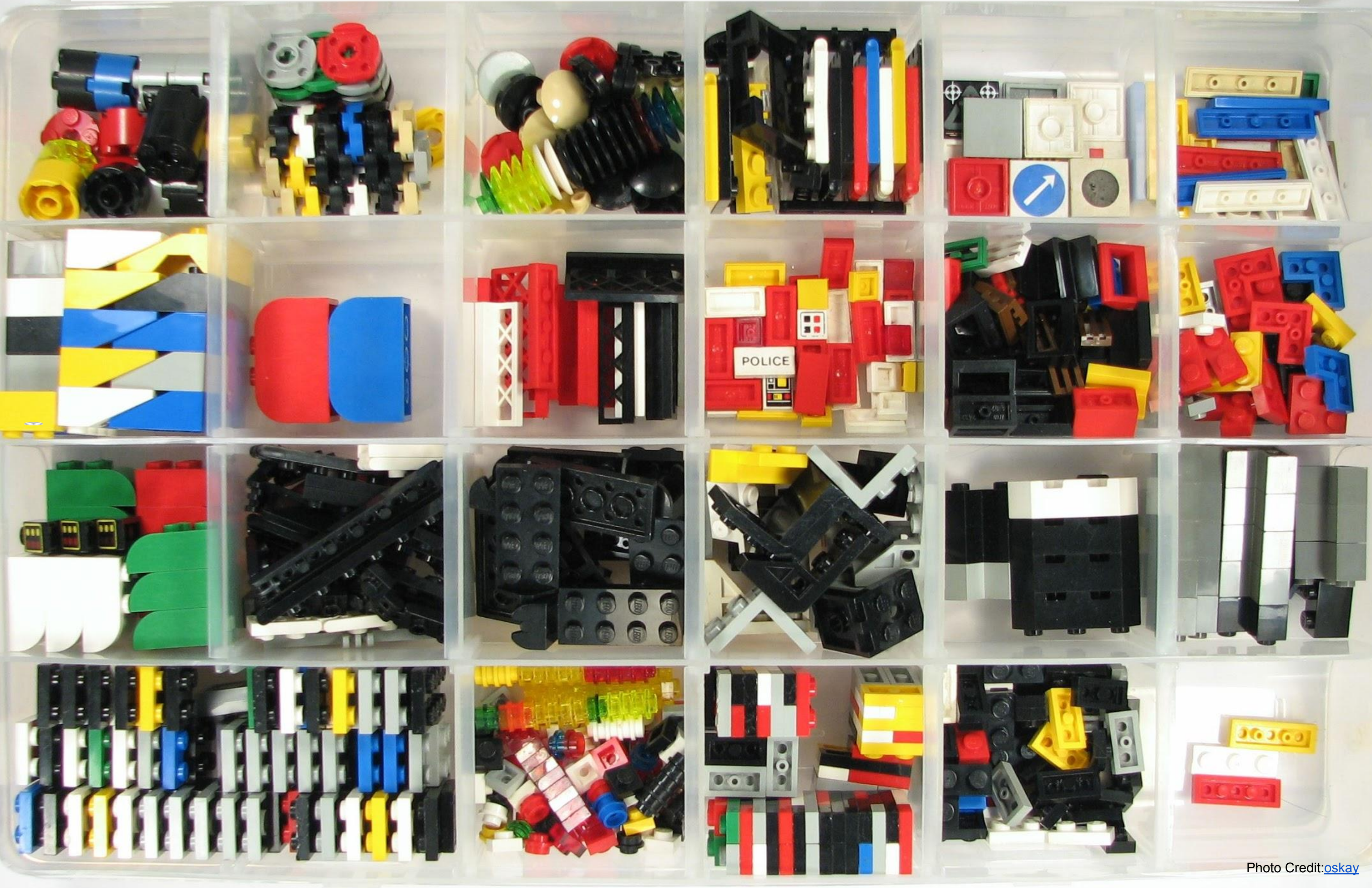


Photo Credit:oskay

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 (future eclipse wakaama)

http://github.com/01org/liblwm2m

Java server implementation

http://github.com/jvermillard/leshan/

Thanks!

More questions? Feel free to contact us!

Matthias Kovatsch

kovatsch@inf.ethz.ch

Julien Vermillard

@vrmvrm

jvermillard@sierrawireless.com