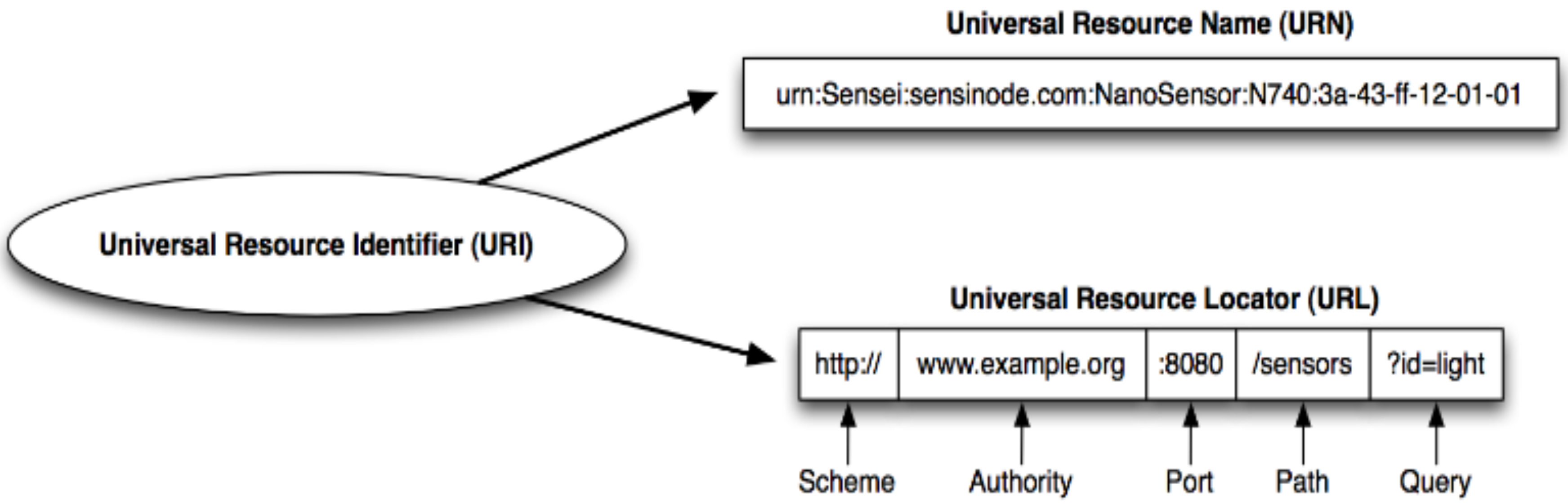
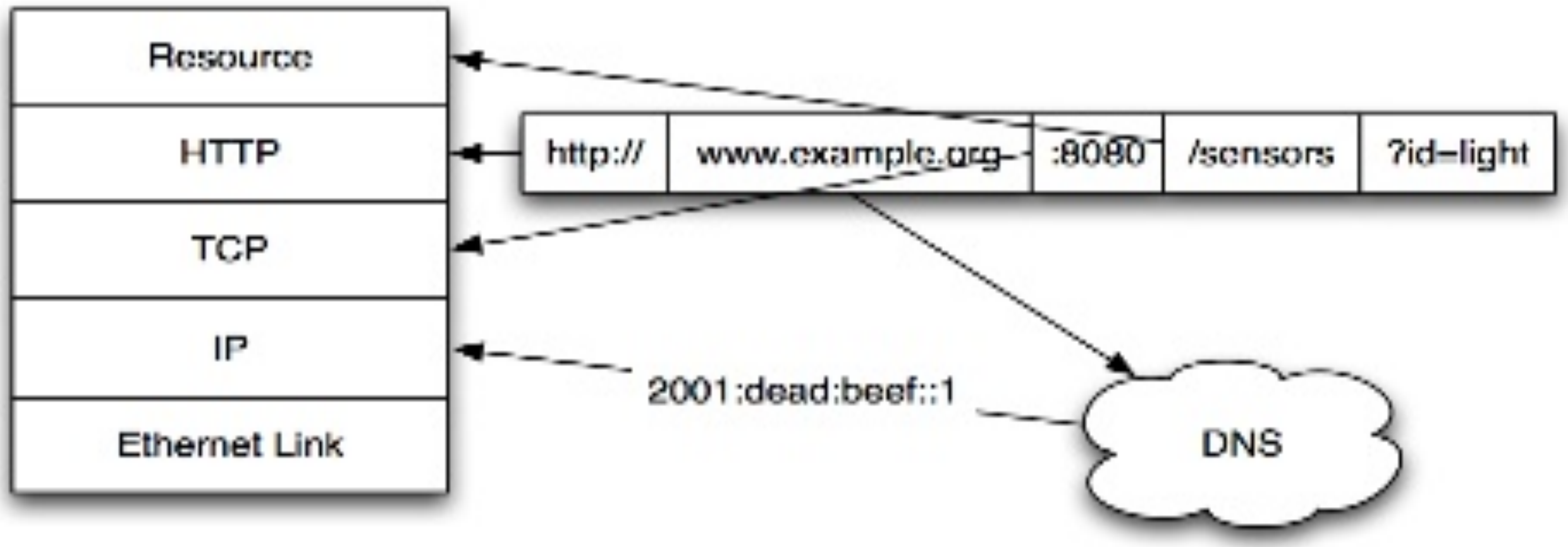
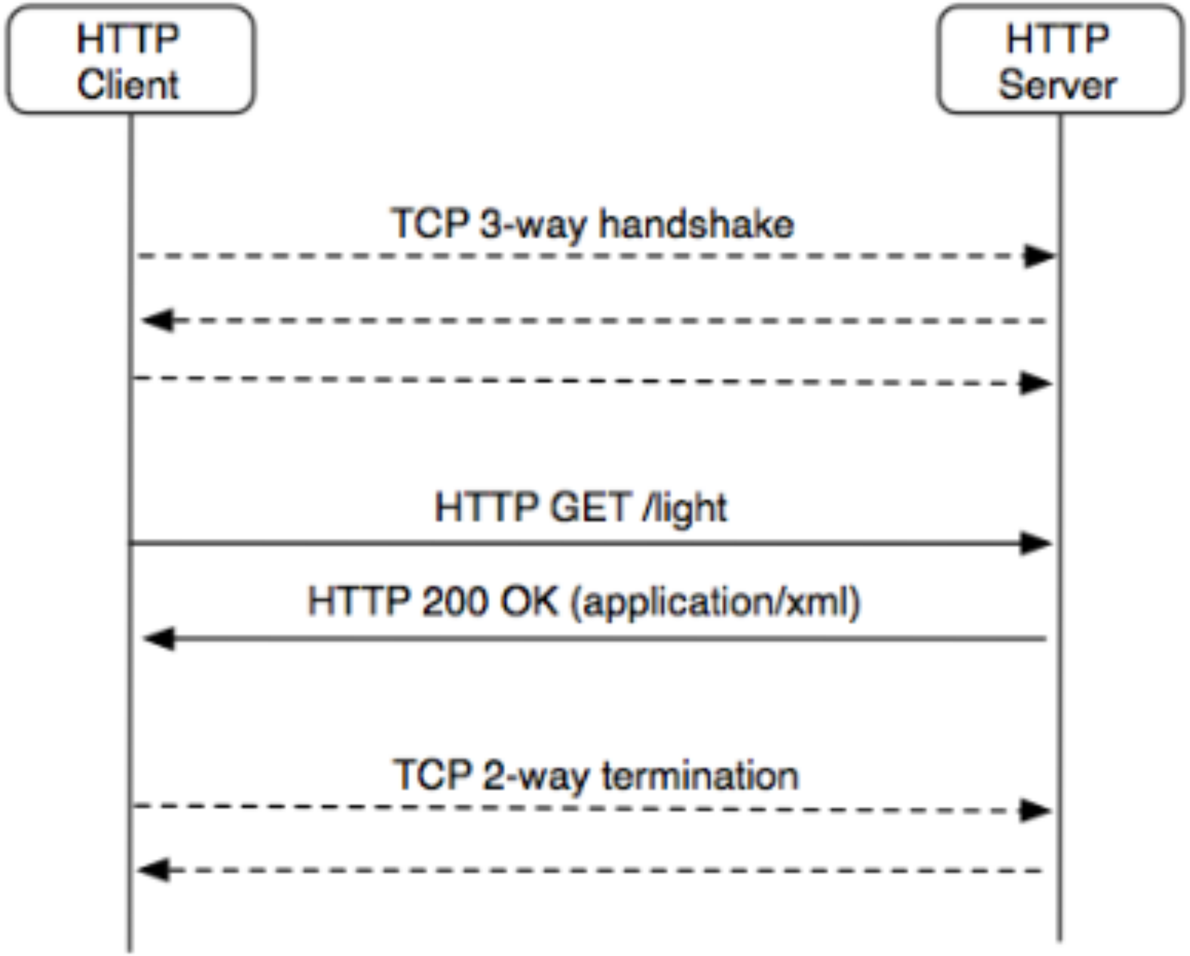
Web resource identification



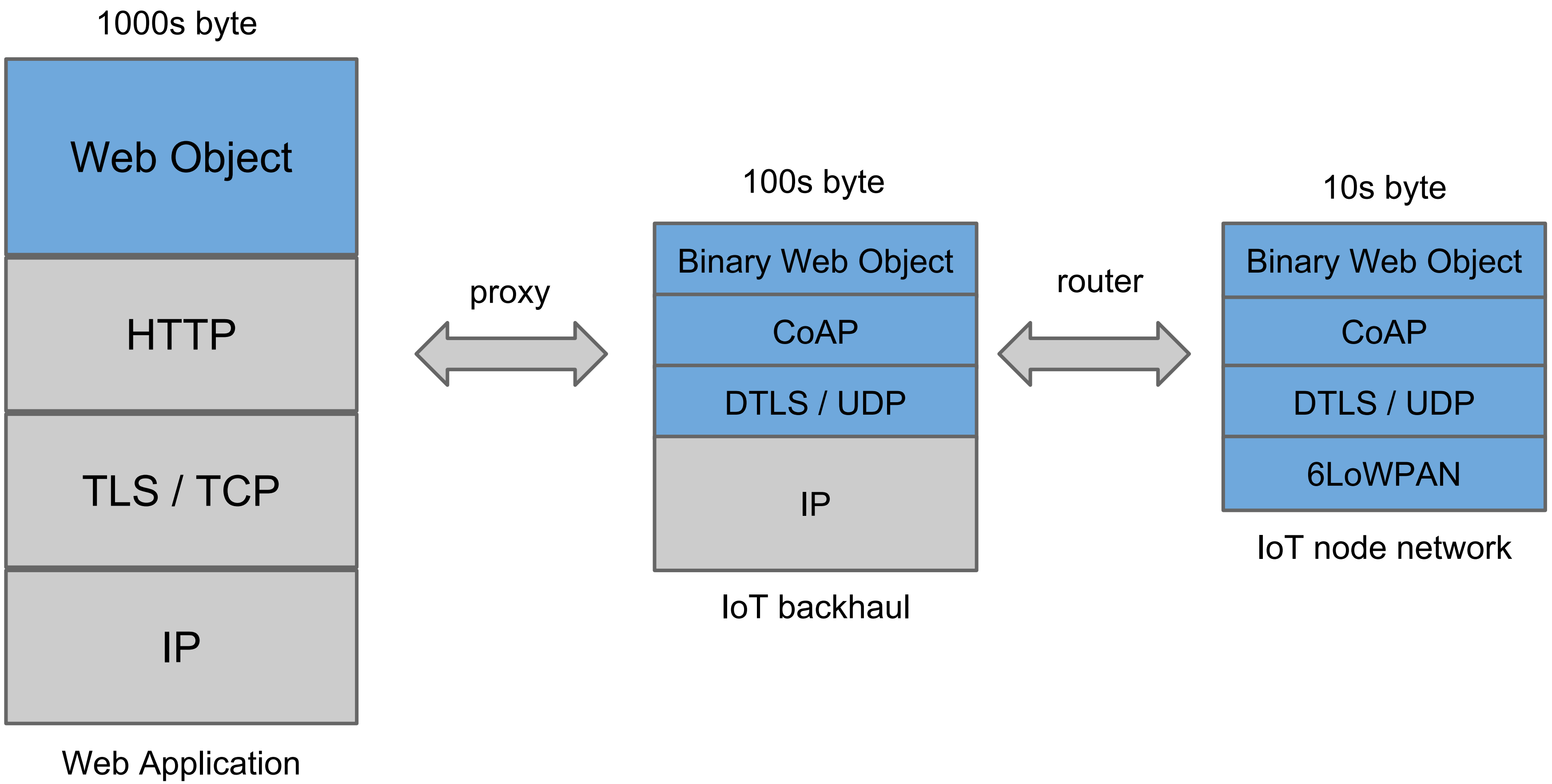
URL Resolution



Example: an HTTP request



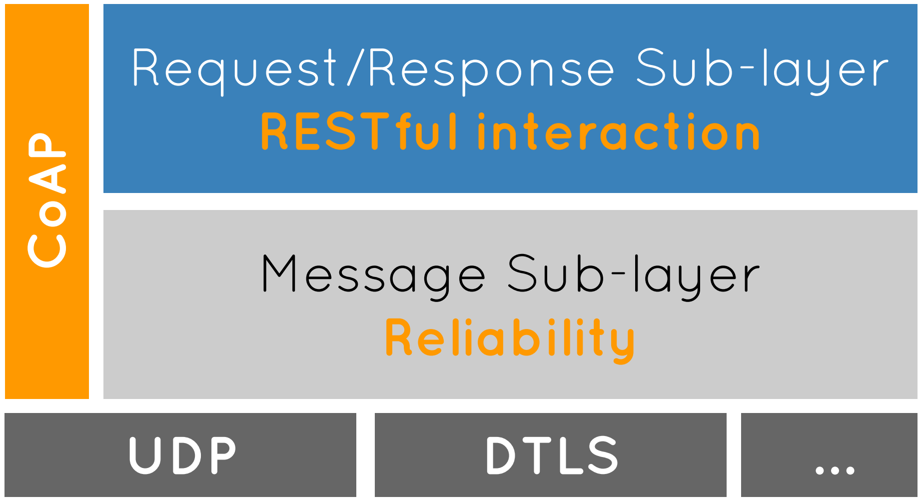
From Web App to IoT nodes



Constrained Application Protocol

CoAP is an application layer protocol tailored for resource constrained devices and M2M applications

- RESTful protocol designed from scratch
- Transparent mapping to HTTP
- Additional features for M2M scenarios (low overhead, multicast support)



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

Deduplication
Optional retransmissions
- -Confirmables "**CON**"
- -Non confirmable "**NON**"
- -Acknoledgment "**ACK**"
- -Reset "**RST**"

Constrained Application Protocol

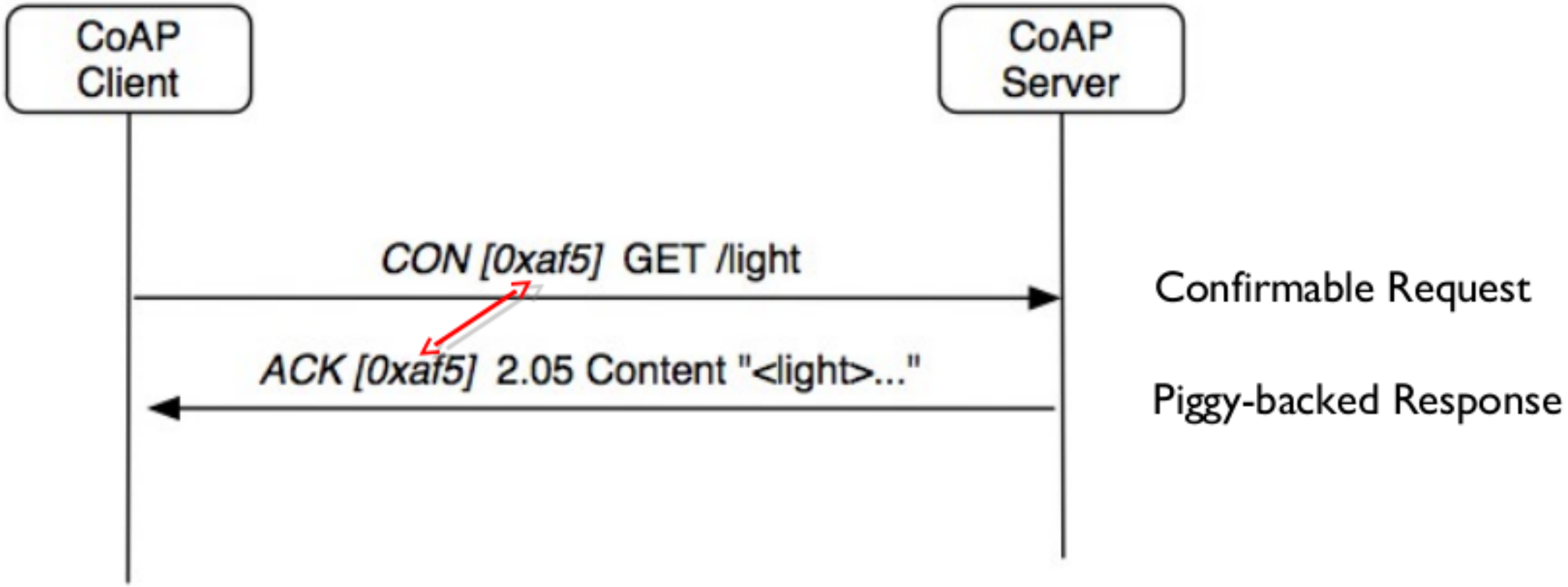
Binary protocol

- Low parsing complexity
- Small message size

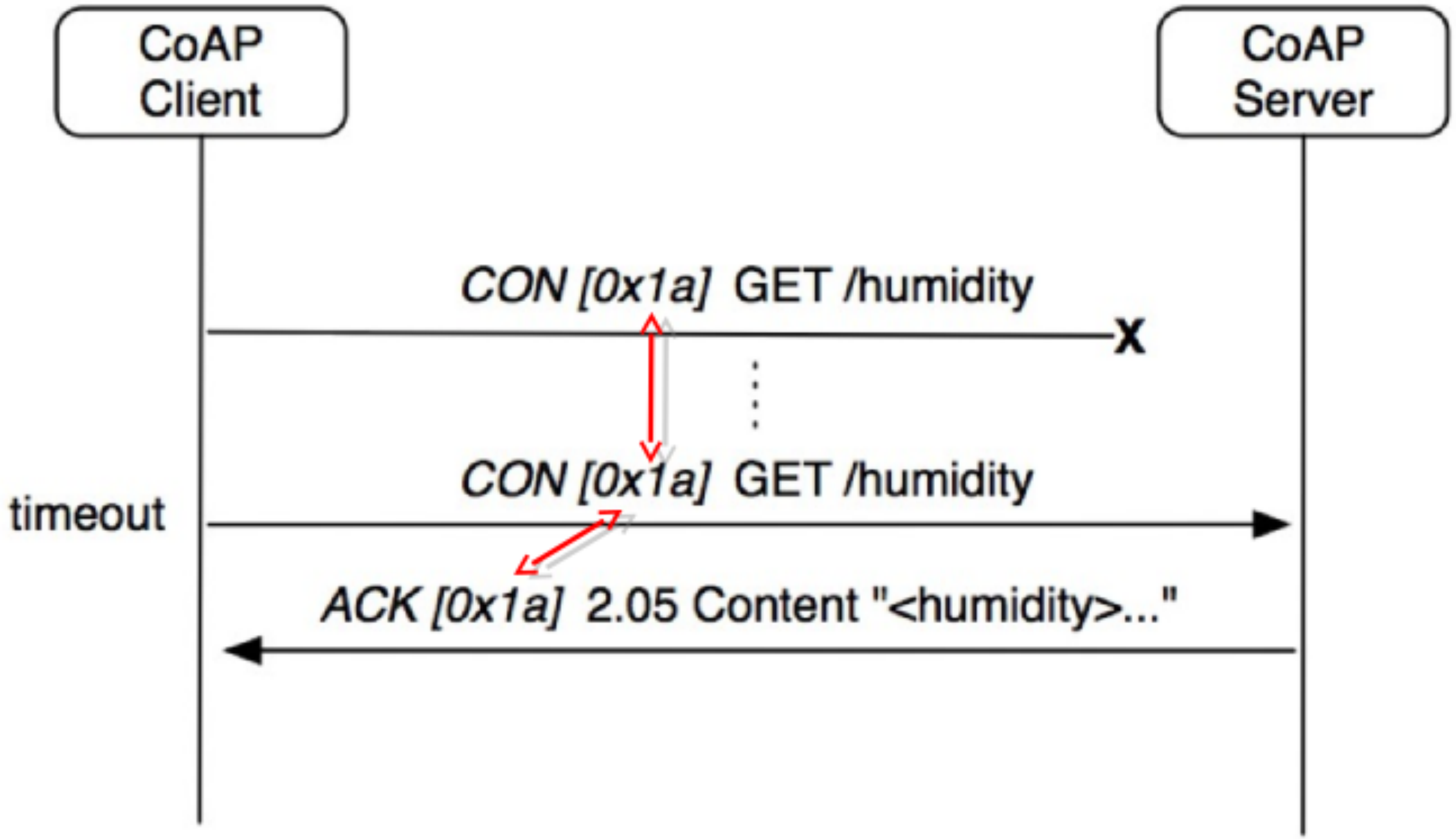
Options

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

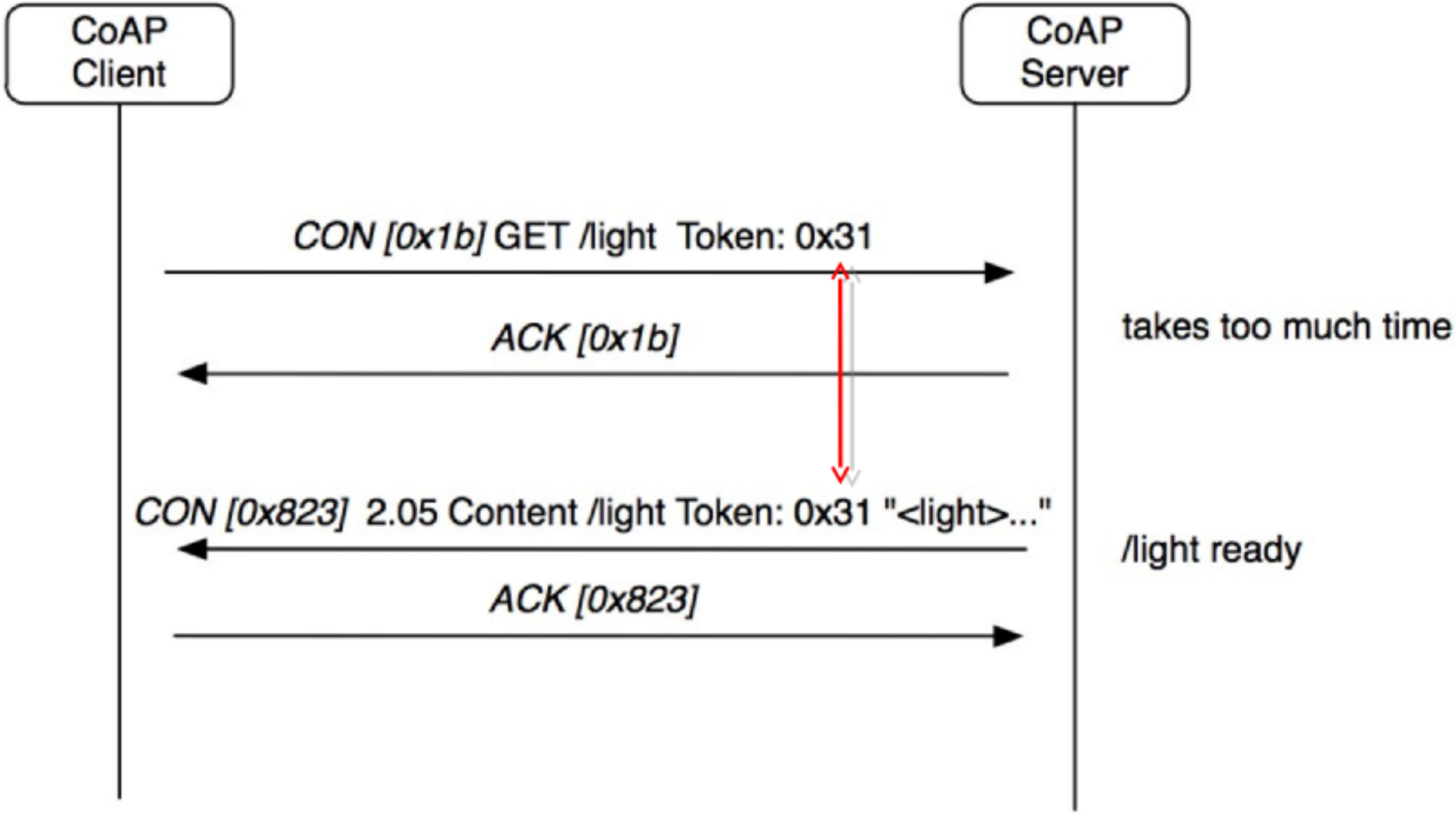
CoAP Request Example



Dealing with packet loss



Separate Response

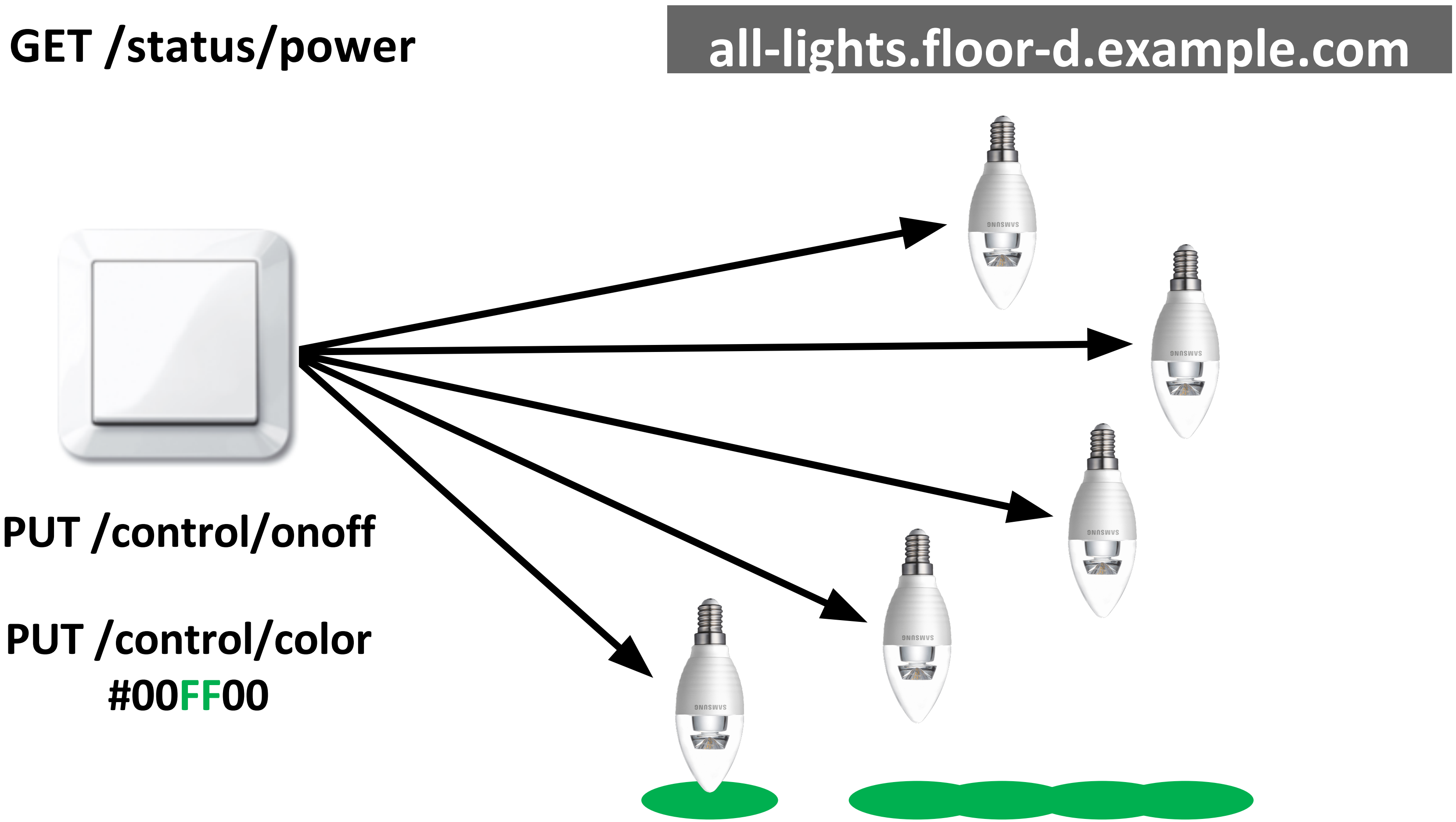


bits and Byte..

```
CLIENT                                                       SERVER
   |                                                            |
   |       ----- CON [0x7d34] GET /temp ------------->          |
   |                                                            |

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 0 |   0   |     GET = 1     |          MID=0x7d34          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   11  |   4   |      "temp" (4 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

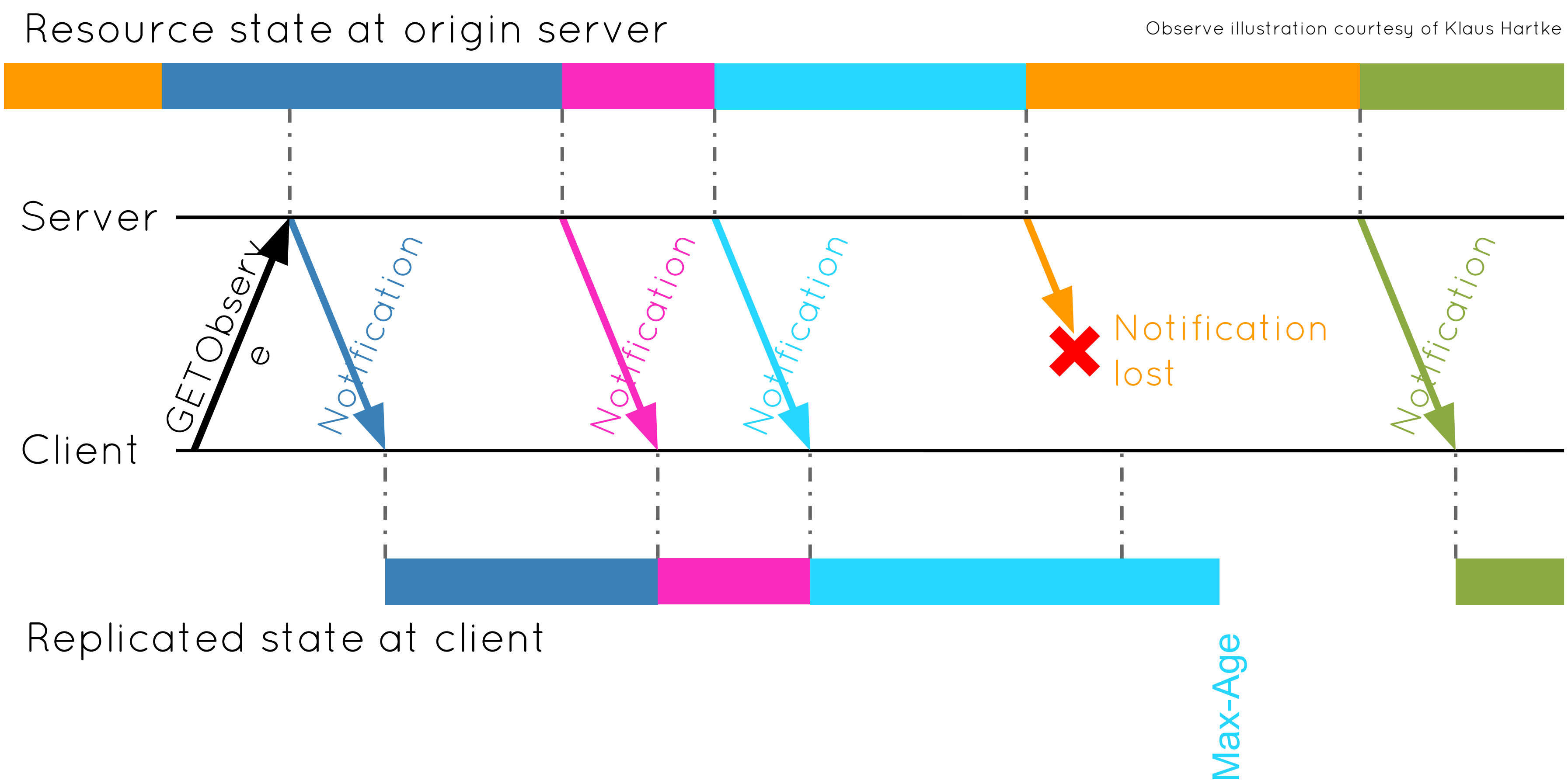
```
CLIENT                                                       SERVER
   |                                                            |
   |     <-------- ACK [0x7d34] 2.05 Content ---------          |
   |                                                            |

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 | 2 |   0   |    2.05=69      |          MID=0x7d34          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     "22.3 C" (6 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

RESTful group communication



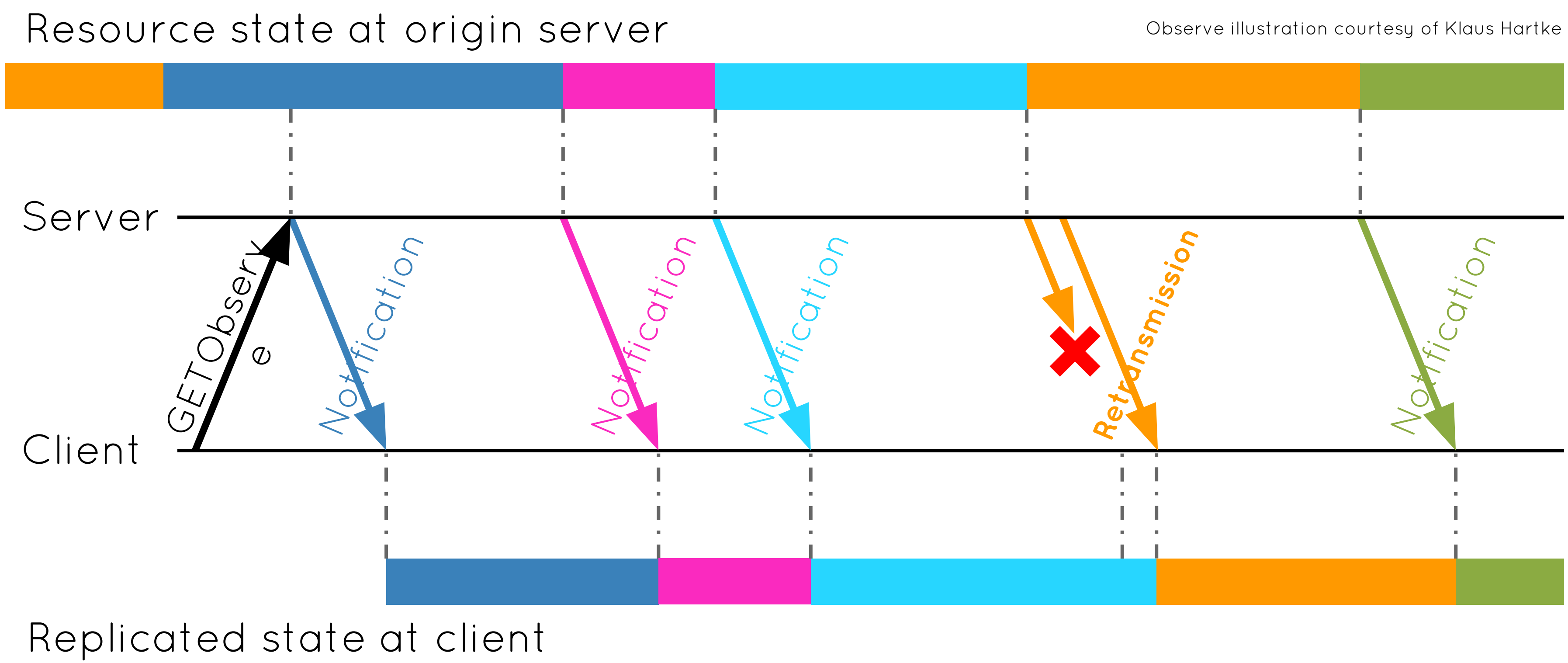
Observing resources - NON mode

current representation of a resource over a period of time



Observing resources - CON mode

current representation of a resource over a period of time



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"
```

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.,

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

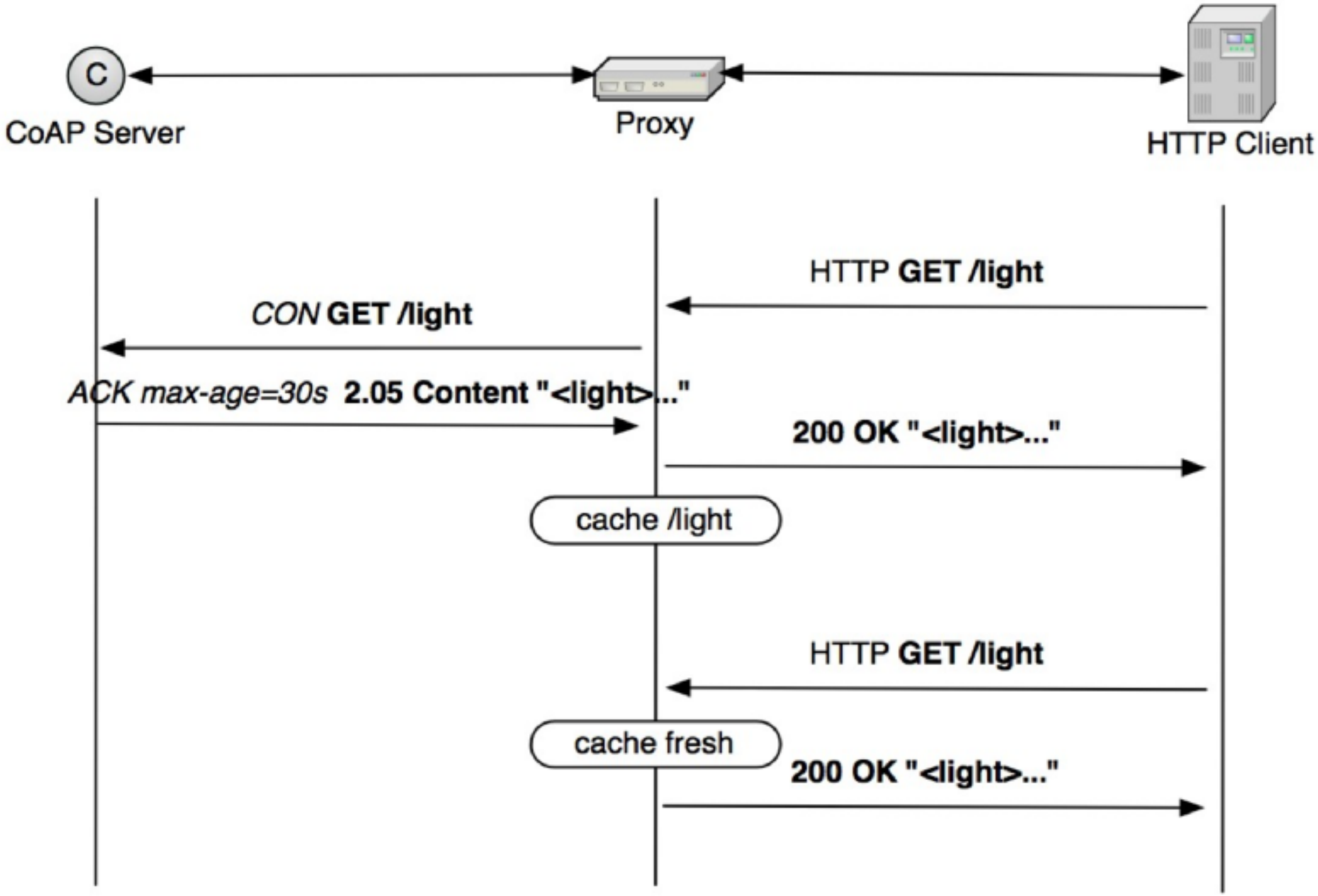
Status of CoAP

In use by

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

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

Proxy


C
CoAP Server
Proxy
HTTP Client
HTTP GET /light
CON GET /light
ACK max-age=30s 2.05 Content "<light>..."
200 OK "<light>..."
cache /light
HTTP GET /light
cache fresh
200 OK "<light>..."

Getting started with CoAP

There are many open source implementations available

- Java CoAP Library Californium
- C CoAP Library Erbium
- libCoAP C Library
- jCoAP Java Library
- OpenCoAP C Library
- TinyOS and Contiki include CoAP support

- CoAP is already part of many commercial products/systems
    - Sensinode NanoService
    - RTX 4100 WiFi Module
- Firefox has a CoAP plugin called Copper
- Wireshark has CoAP dissector support
- Implement CoAP yourself, it is not that hard!

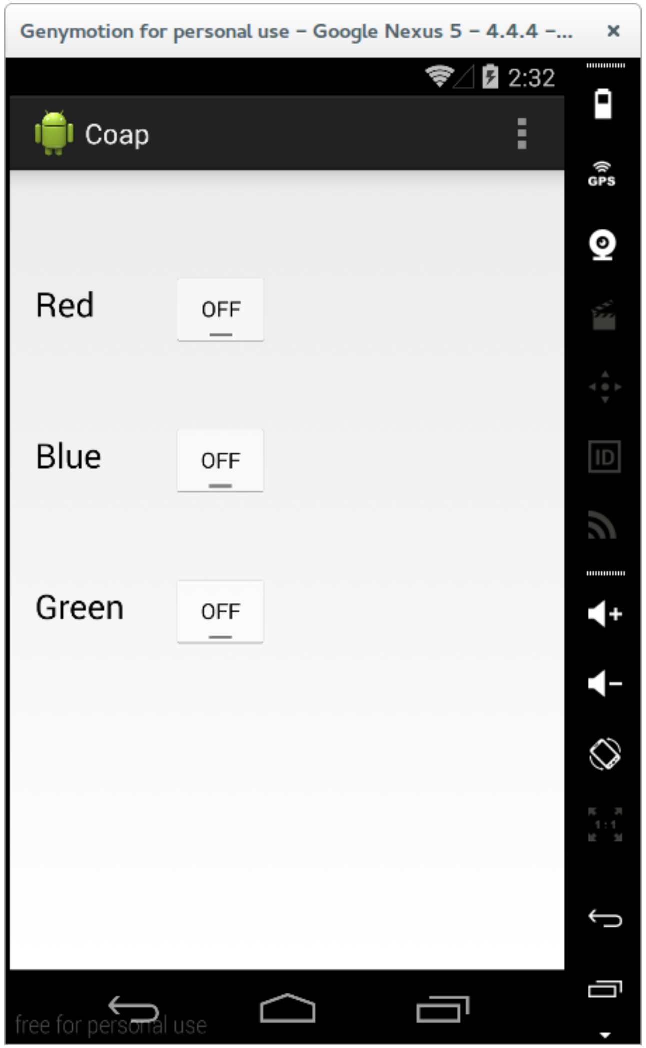
Using CoAP on Android

1. Create POST requests
2. Use POST method on leds
3. Send HTTP request to the proxy
4. Turn on/off TelosB leds

Proxy: http://192.168.0.10:8080/proxy/

Coap: coap://[aaaa::212:7400:1024:1bf3]:5683/actuators/leds?color=r/b/g

POST payload: mode=on/off



Some hints

<uses-permission android:name="android.permission.INTERNET" />

USE: HttpClient and HttpPost

post.setEntity(new UrlEncodedFormEntity(*pairs*));

Payload POST (e.g., MODE=ON/OFF)

pairs can be a list<NameValuePair> or JSONObject

You should use threads to handle communication:

new Thread(new Runnable(){**your code**}).start

Finally you should execute your HttpPost on your HttpClient something like: client.execute(post);

Wireless Sensor Network

Made of motes:

–Battery powered (e.g., 2xAA alkaline)

–Wireless (2.4Ghz, 868Mhz, 433Mhz,...)

–Limited computational power

–Limited memory

–Communication interface

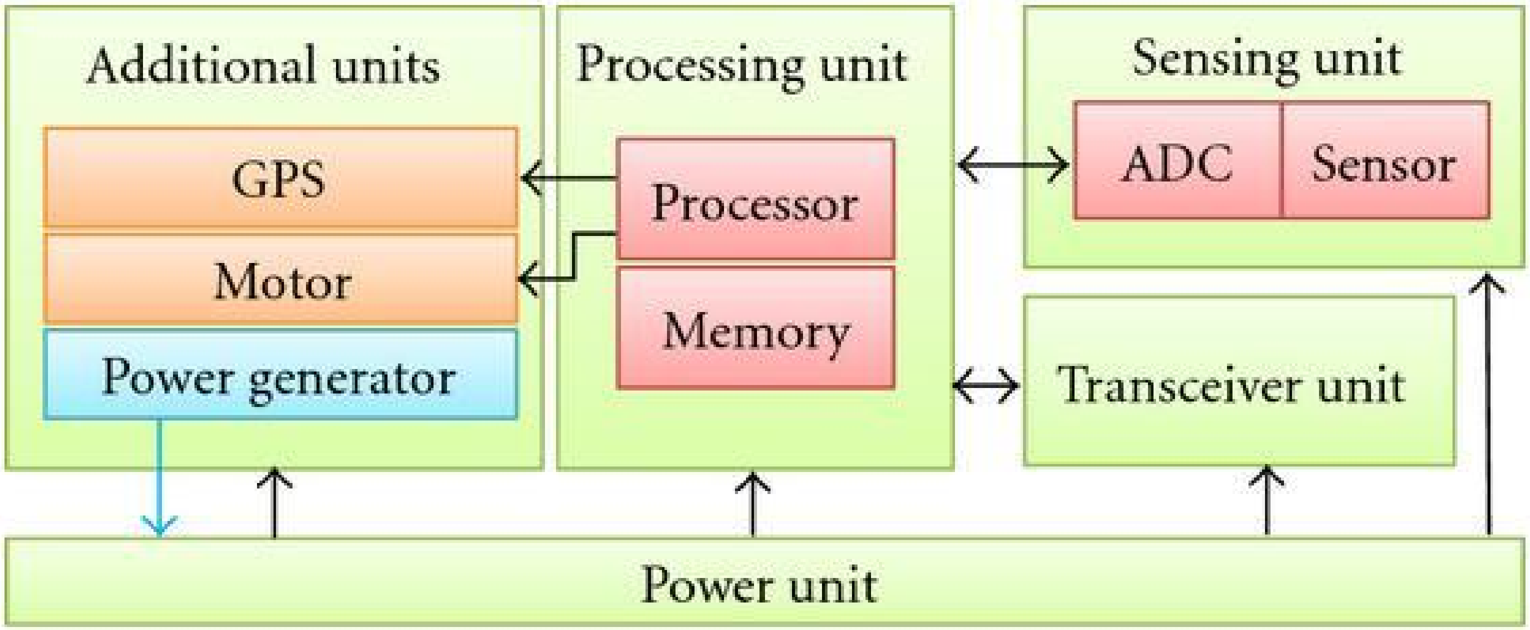
} Microcontroller (MCU)

•External transducers (e.g., temperature, humidity, pressure,...)

•External transceivers

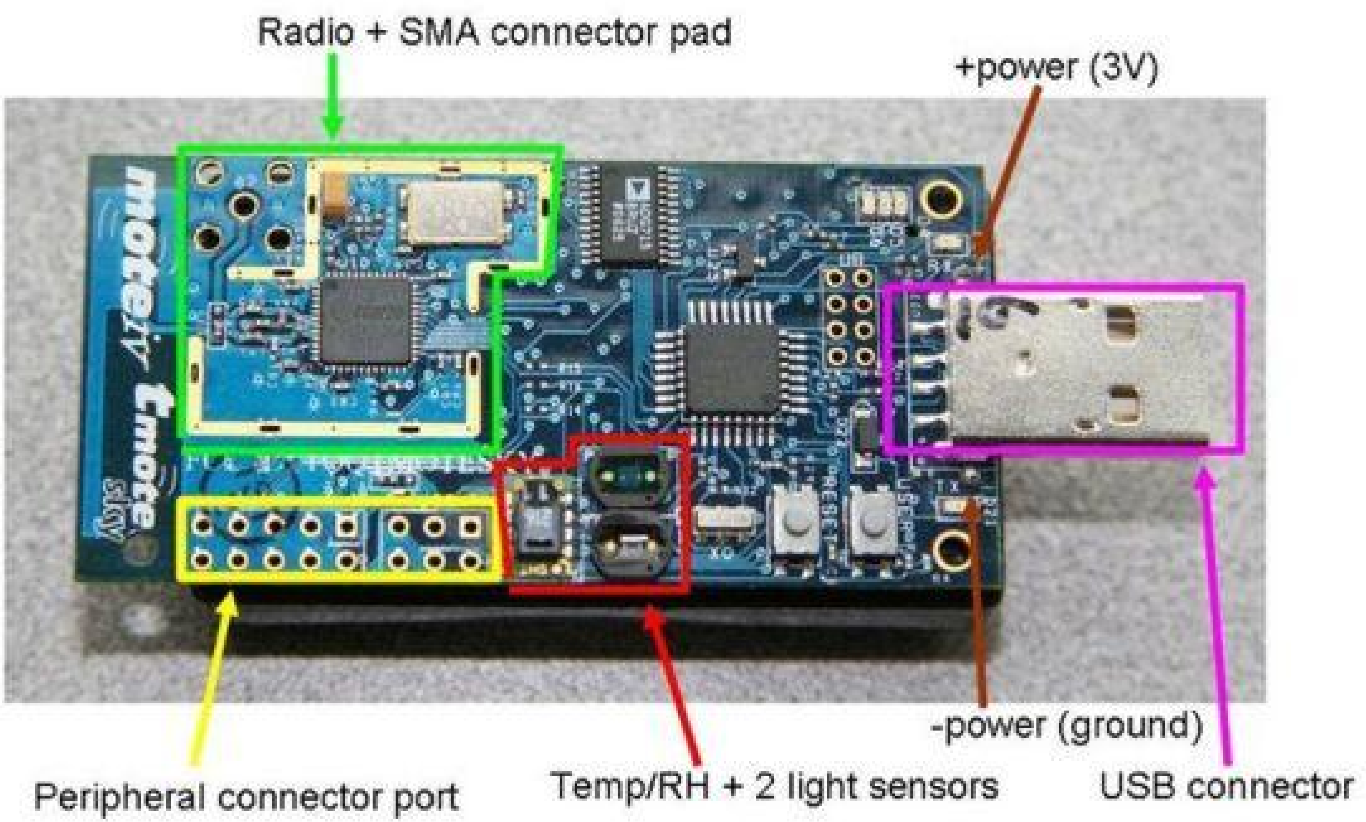
•External flash memory

WSN motes architecture



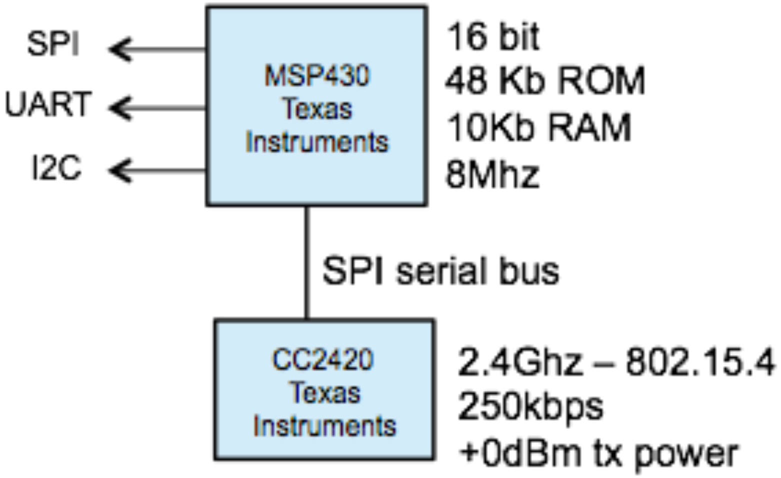
GPS: global positioning system
ADC: analog to digital converter

Example: Telos B mote

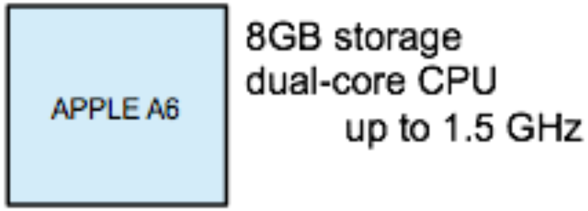


Hardware characteristics

TelosB



iPhone 5c



Energy consumption

iPhone 5C

- Power: Non-removable Li-Po
- Expensive!
- Battery lasts < 1 day

TelosB

- Power: 2xAA alkaline batteries
- Cheap!
- With power management, battery lasts up to a couple of years

TinyOS

- Open-source OS for embedded systems
- Open-source
  - Source code easily reusable
  - Large developers community
- Very small, flexible “programming framework”
  - Less than 400 bytes!
- Written in NesC (a dialect of C)

Why a dedicated operating system?

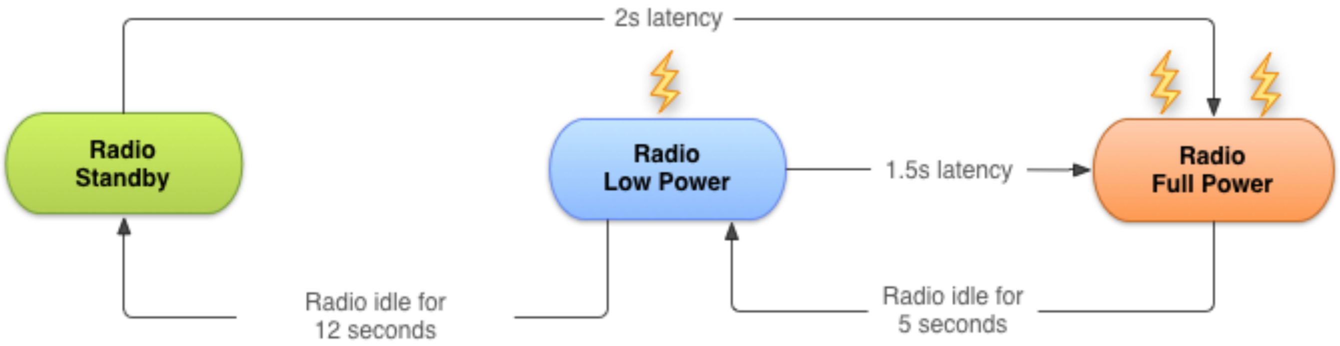
- Motes are intentionally tiny!
  - 1-MIPS processor
  - 10KB of storage
  - Low power
- Special needs for sensor networks
  - Real-time
  - Reactive concurrency
  - Flexibility

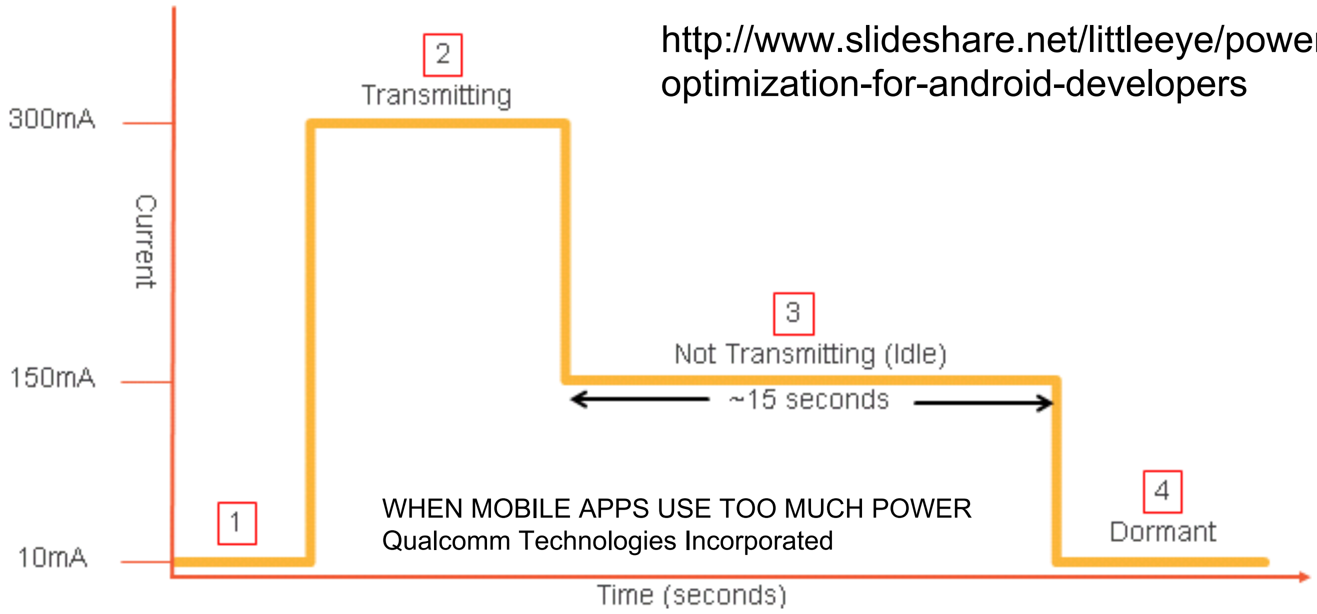
TinyOS

Native support for low-power operations:

- Microcontroller Power Management
  - Microcontrollers should always be in the lowest power state possible
  - TinyOS handles state transitions automatically to achieve maximum power saving
- Radio Power Management
  - Duty-cycle radio to save energy and extend network lifetime
- Peripheral Energy Management
  - Energy-efficient scheduling of sensing operation and peripheral access

Mobile network states



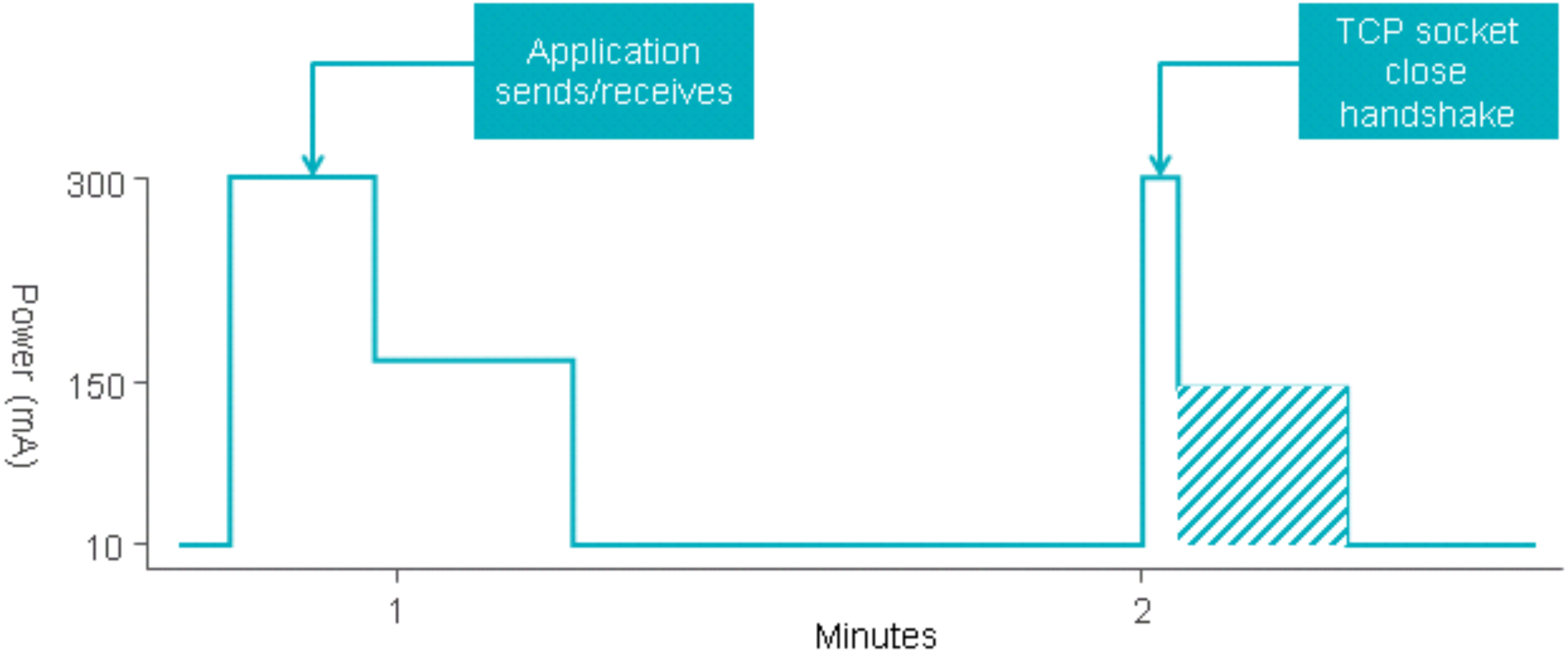


http://www.slideshare.net/littleeye/power-optimization-for-android-developers

WHEN MOBILE APPS USE TOO MUCH POWER
Qualcomm Technologies Incorporated

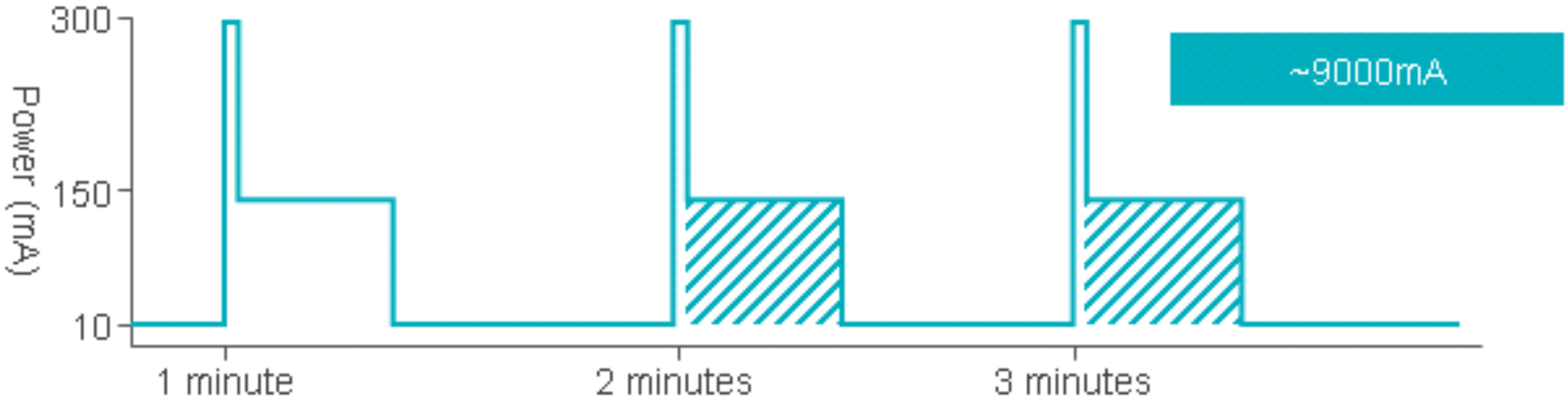
Hanging socket

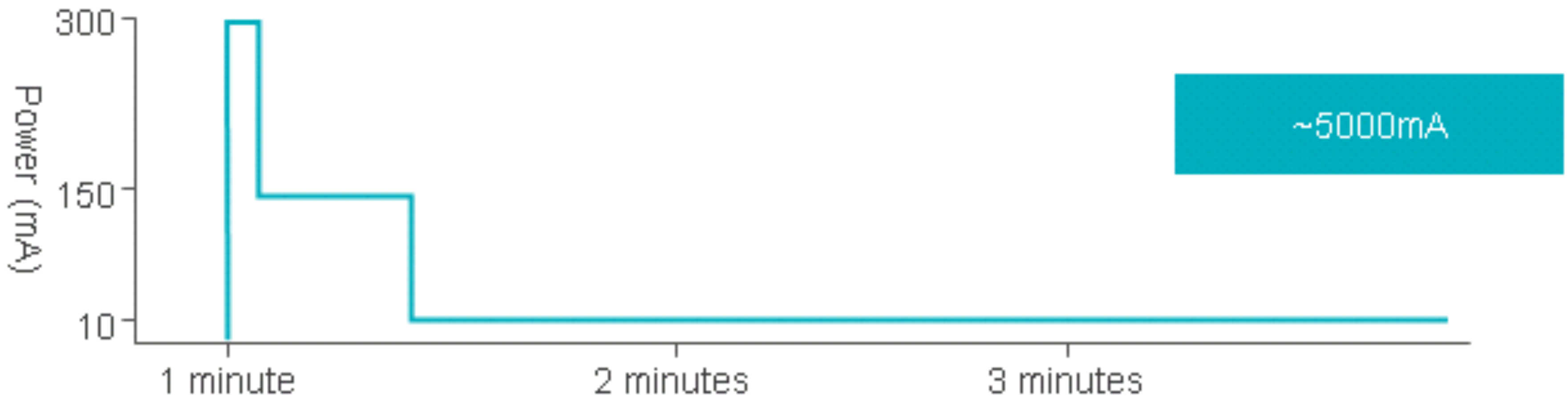
- Program your apps to close sockets when done



If network is used 4 times per hour -> 10 hours of standy instead of 8 (20% improvement!)

Ungrouped vs grouped network activity





Projects

- Build your own walking detector app:
  - Use one or more low-power sensors (sensors fusion)
  - Experiment with different features
  - Collect your own training dataset (friends can help)
  - Test it!
- Build an Indoor Localization app
  - RSSI
  - DR
  - Try some filter e.g., EKF for sensor fusion and navigation
  - Acoustic
- CoAP and sensors
  - TelosB + Android Localization