

ARM IoT Tutorial

CoAP: The Web of Things Protocol

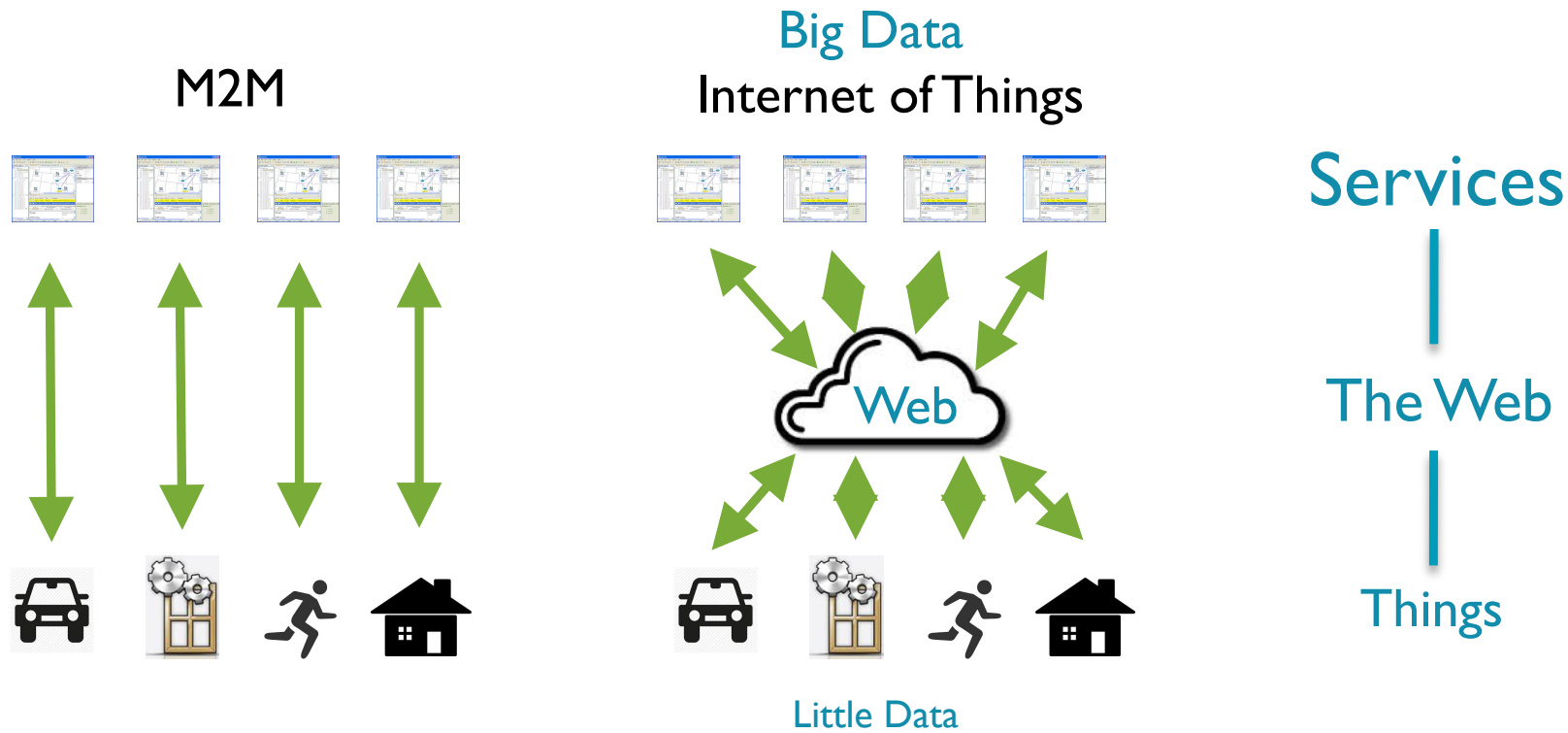
Zach Shelby

April 30th, 2014



Introduction

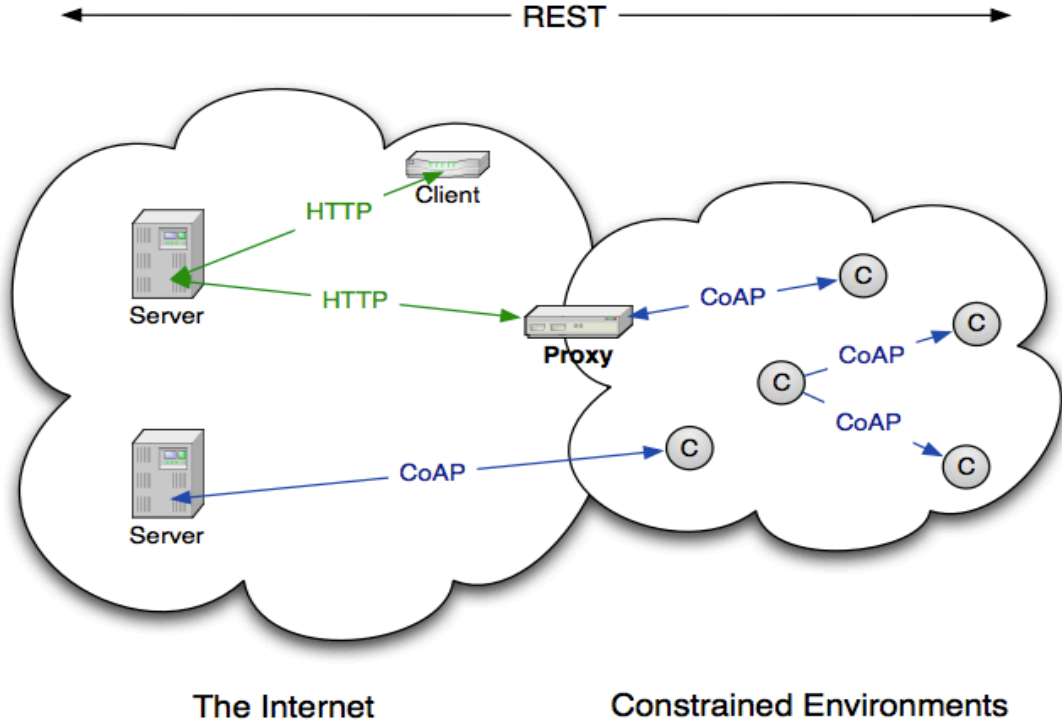
Evolution from M2M to IoT



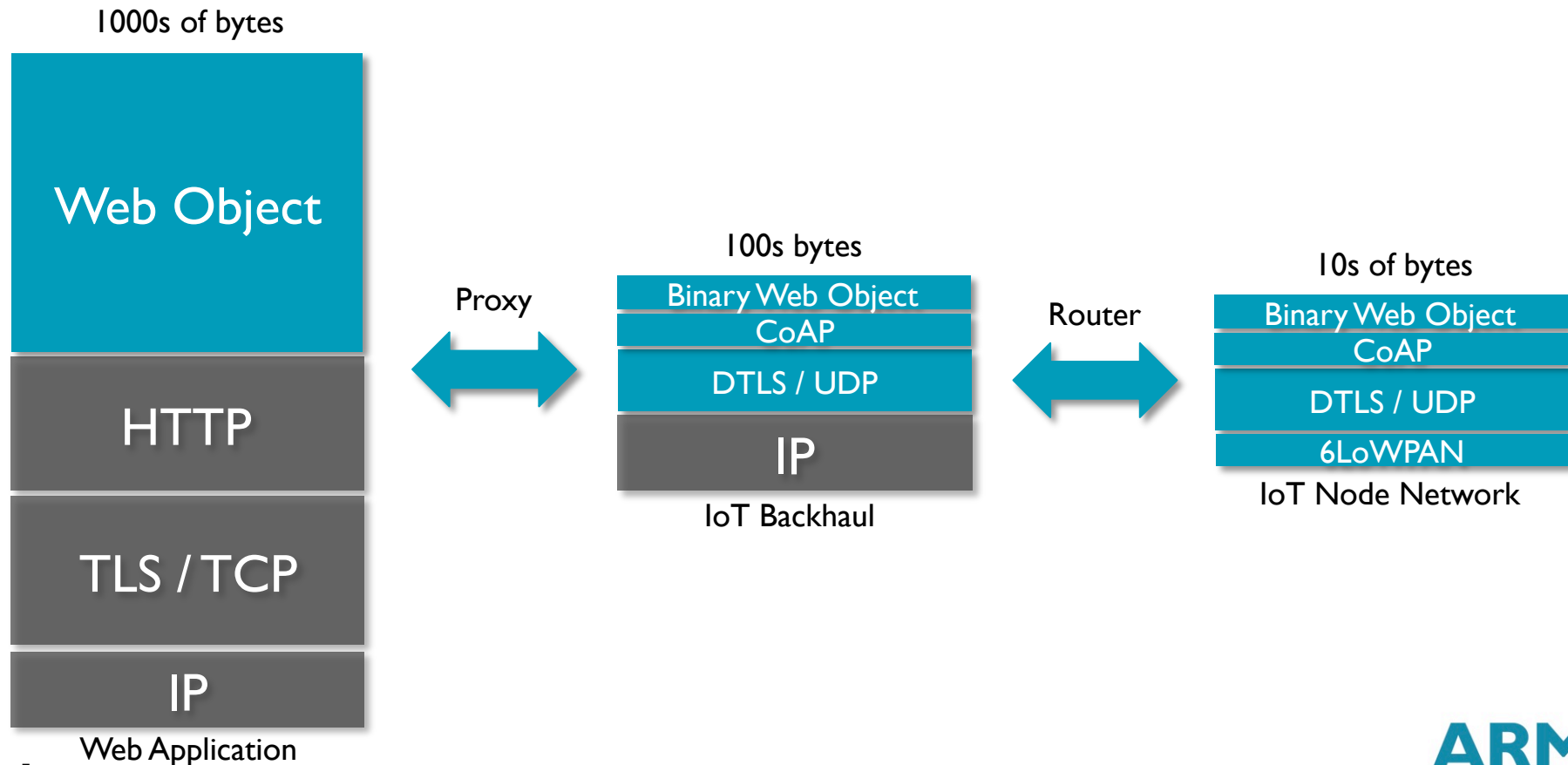
CoAP: The Web of Things Protocol

- Open IETF Standard
- Compact 4-byte Header
- UDP, SMS, (TCP) Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-in Discovery

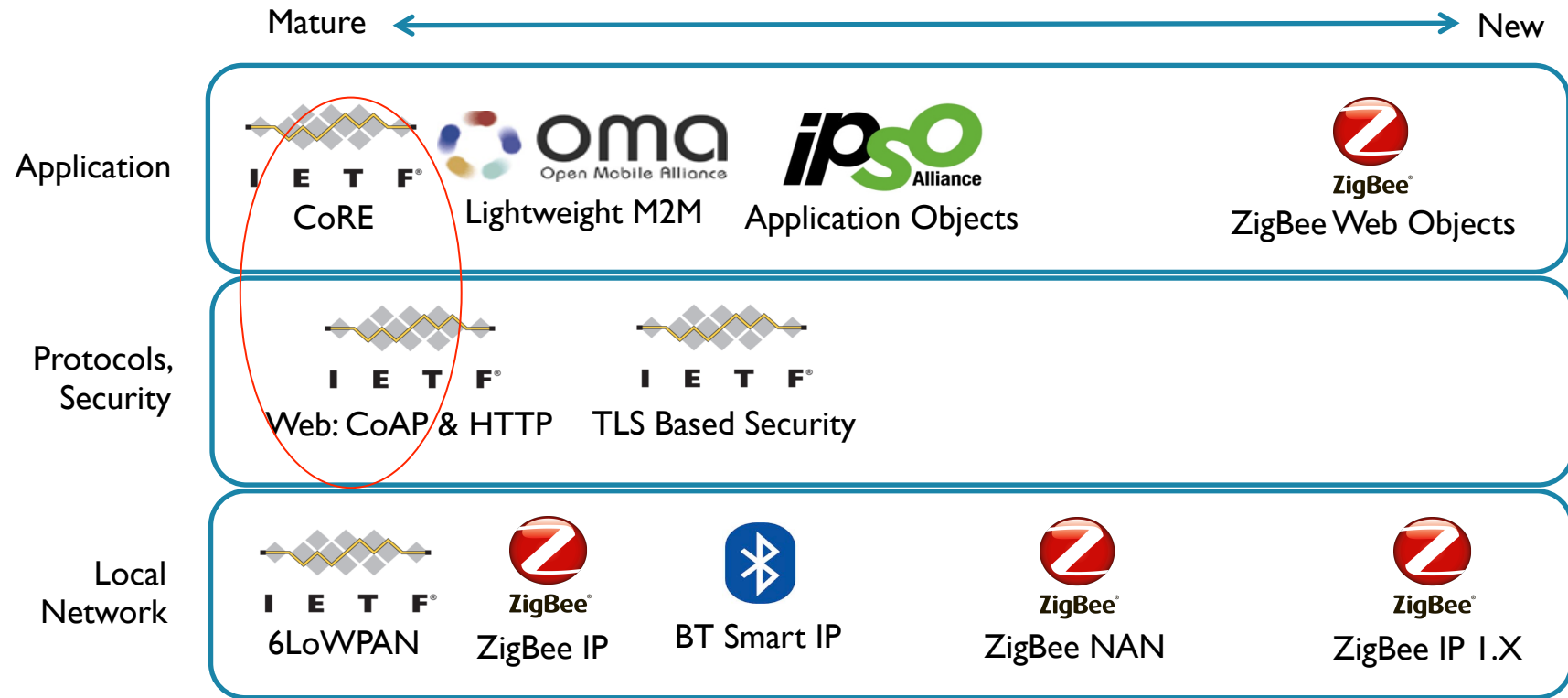
CoAP	
DTLS	SMS
UDP	
IP	



From Web Applications to IoT Nodes

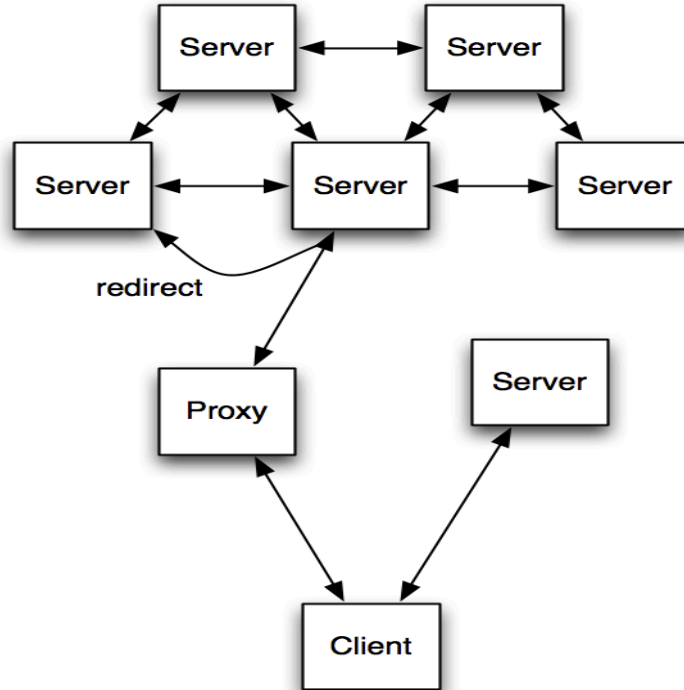


CoAP is One Key IoT Standard

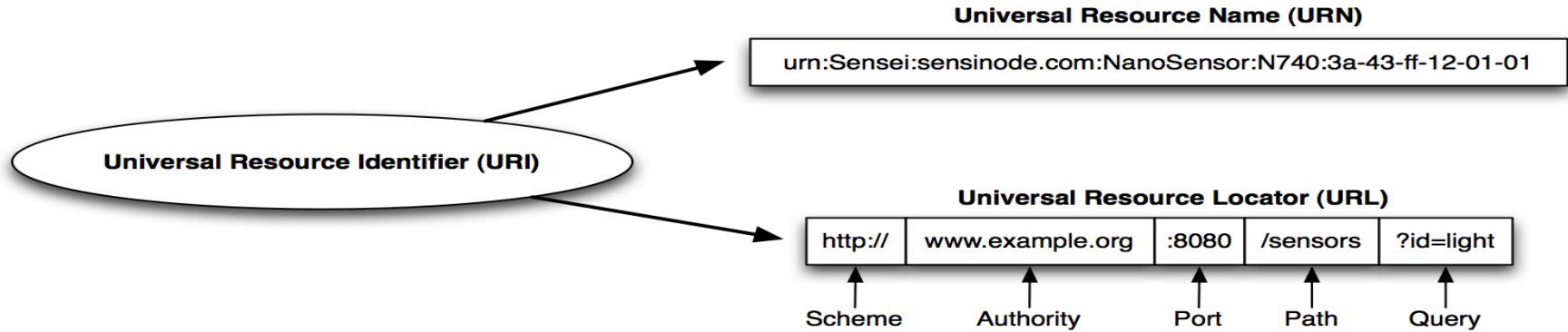


The Web and REST

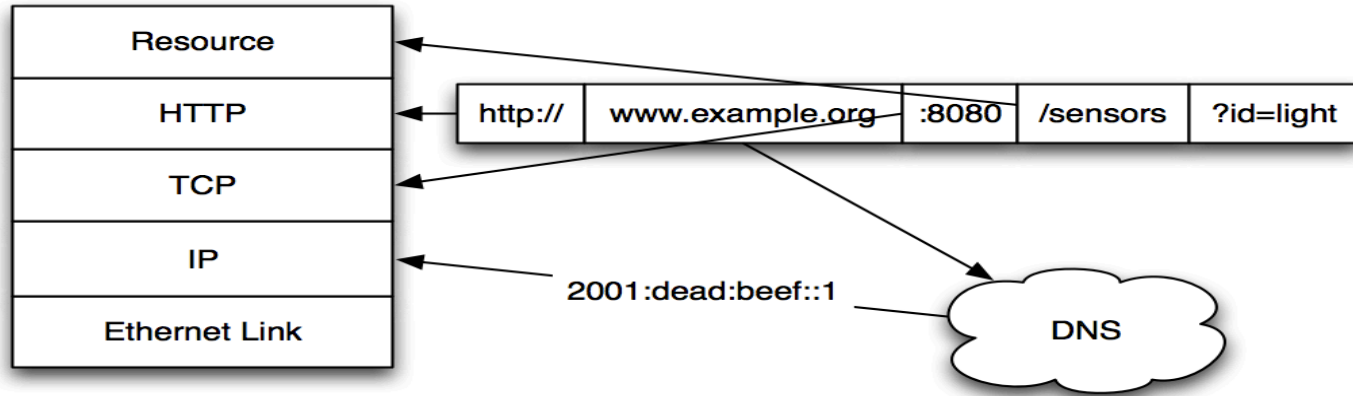
The Web Architecture



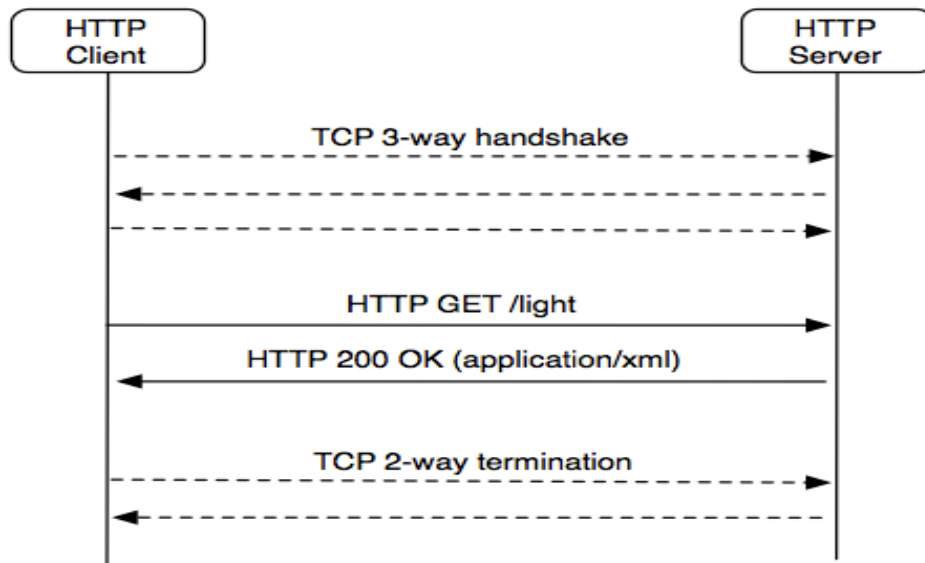
Web Naming



URL Resolution

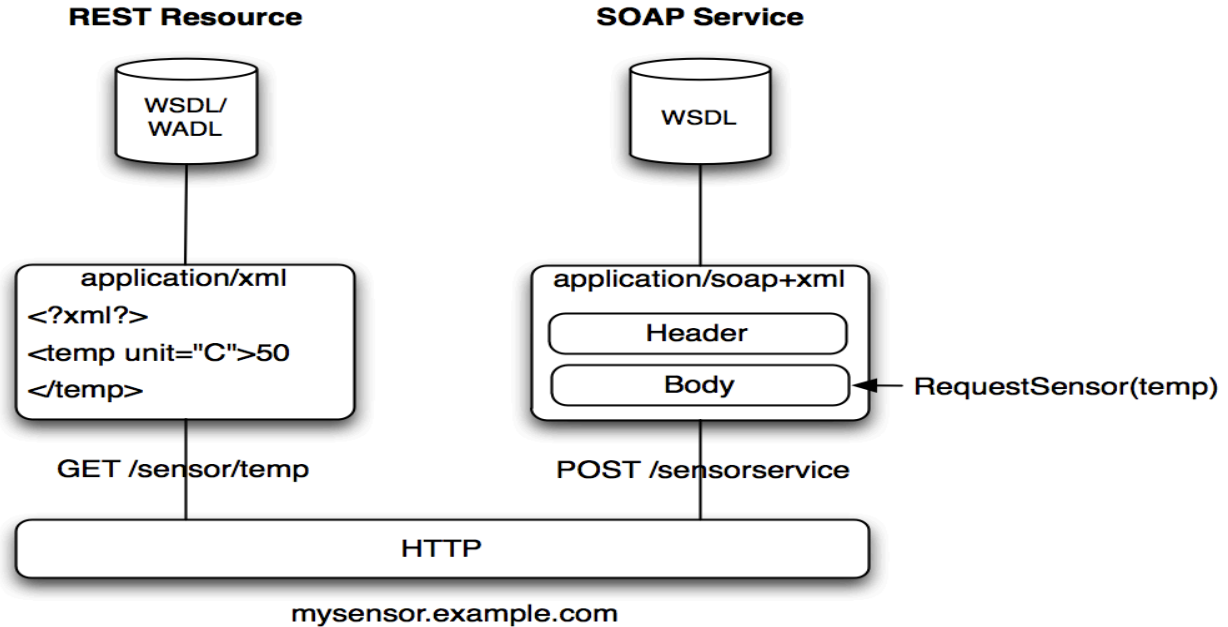


An HTTP Request

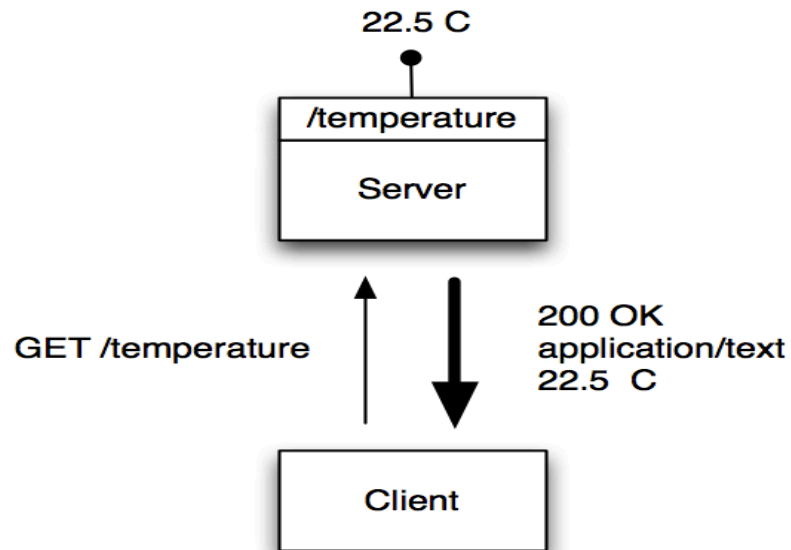


See RFC2616 - Hypertext Transfer Protocol v1.1

Web Paradigms

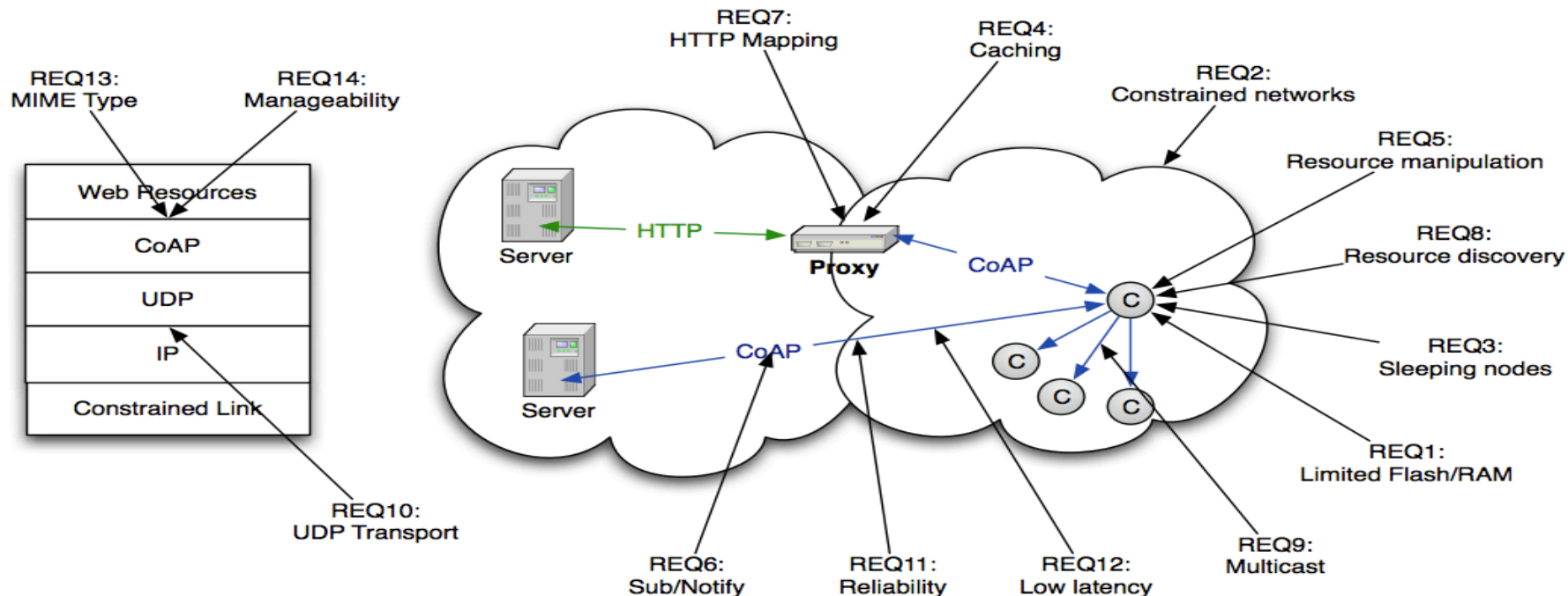


A REST Request

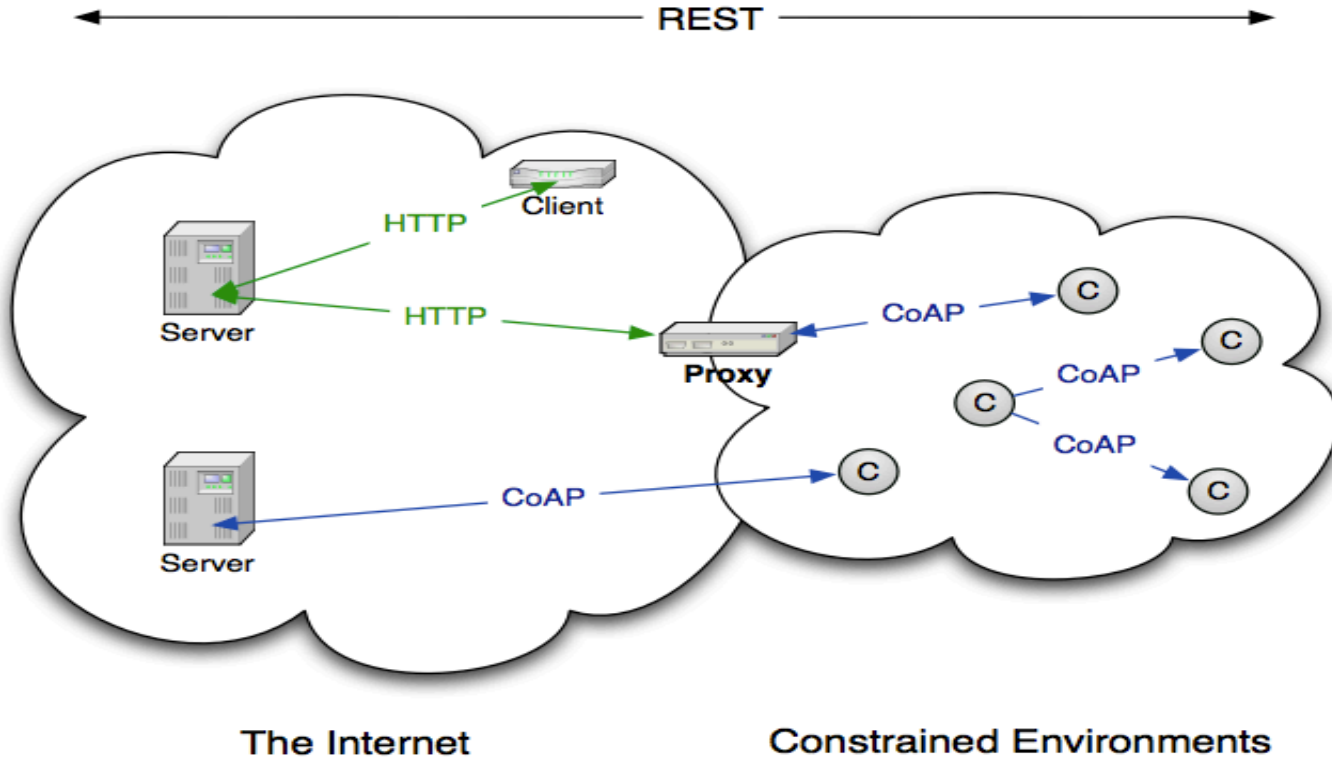


CoAP: Constrained Application Protocol

CoAP Design Requirements



The CoAP Architecture



What CoAP is (and is not)

- Sure, CoAP is
 - A very efficient RESTful protocol
 - Ideal for constrained devices and networks
 - Specialized for M2M applications
 - Easy to proxy to/from HTTP
- But hey, CoAP is not
 - A general replacement for HTTP
 - HTTP compression
 - Restricted to isolated “automation” networks

CoAP Features

- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS based PSK, RPK and Certificate security
- Subset of MIME types and HTTP response codes
- Built-in discovery
- Optional observation and block transfer

Transaction Model

- Transport

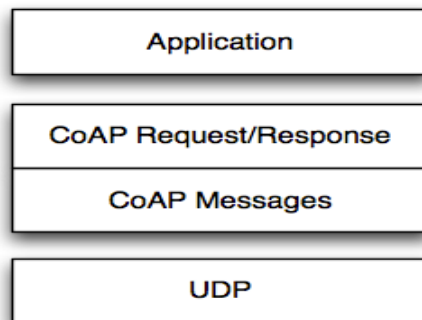
- CoAP currently defines:
 - UDP binding with DTLS security
 - CoAP over SMS or TCP possible

- Base Messaging

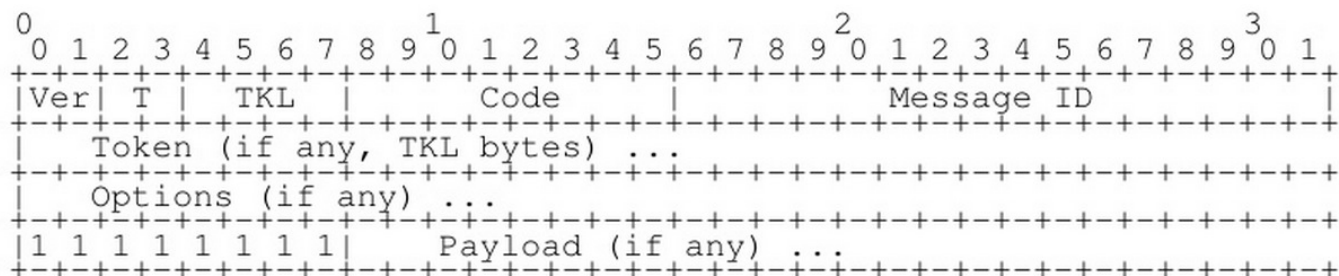
- Simple message exchange between endpoints
- Confirmable or Non-Confirmable Message answered by Acknowledgement or Reset Message

- REST Semantics

- REST Request/Response piggybacked on CoAP Messages
- Method, Response Code and Options (URI, content-type etc.)



Message Header (4 bytes)



Ver - Version (1)

T - Message Type (Confirmable, Non-Confirmable, Acknowledgement, Reset)

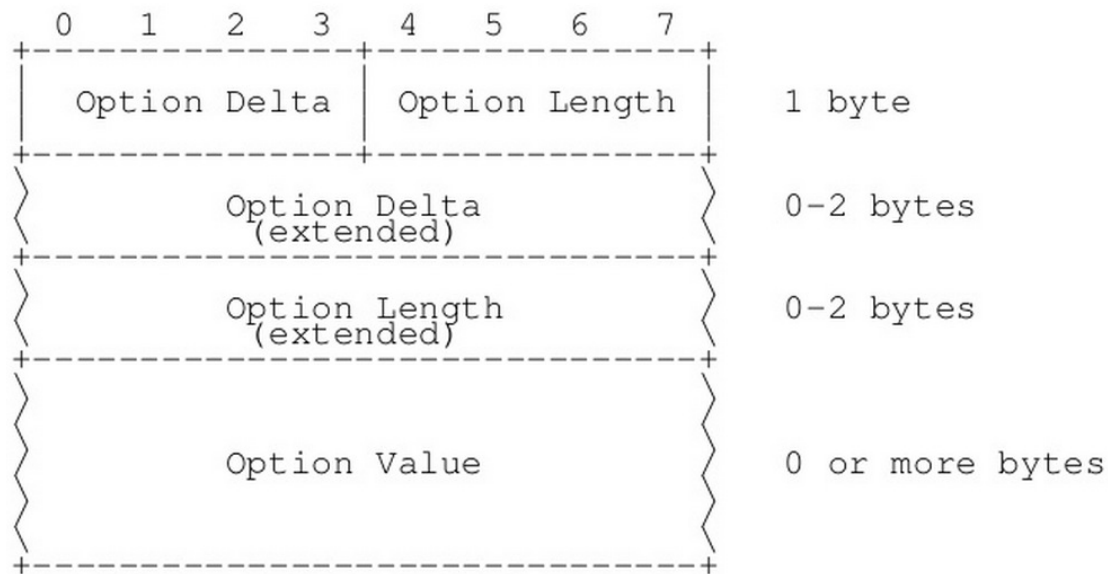
TKL - Token Length, if any, the number of Token bytes after this header

Code - Request Method (1-10) or Response Code (40-255)

Message ID - 16-bit identifier for matching responses

Token - Optional response matching token

Option Format



Option Delta - Difference between this option type and the previous

Length - Length of the option value

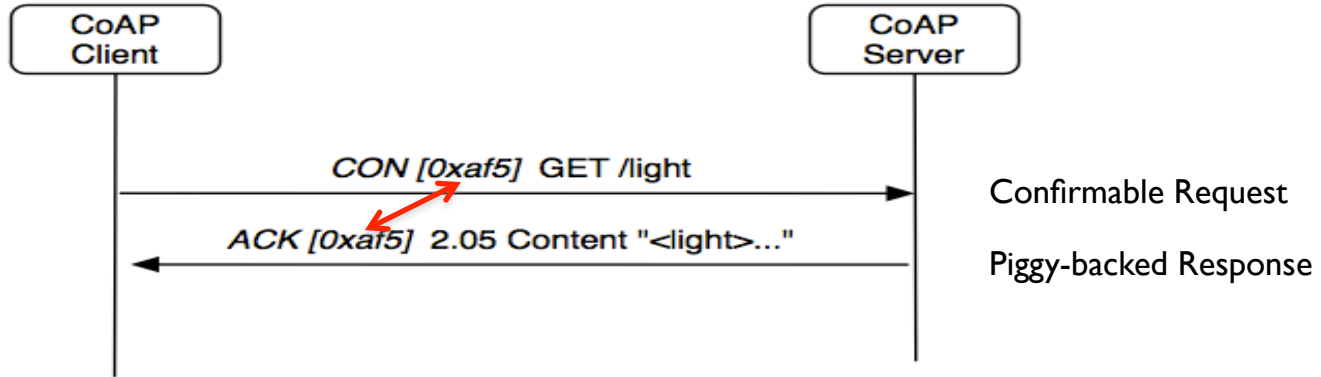
Value - The value of Length bytes immediately follows Length

Base Specification Options

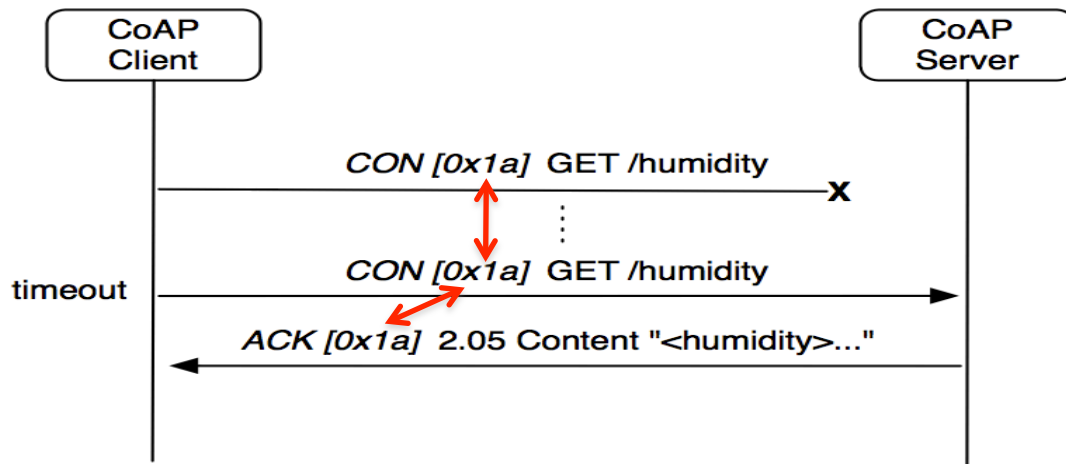
No.	C	U	N	R	Name	Format	Length	Default
1	x			x	If-Match	opaque	0-8	(none)
3	x	x	-		Uri-Host	string	1-255	(see below)
4				x	ETag	opaque	1-8	(none)
5	x				If-None-Match	empty	0	(none)
7	x	x	-		Uri-Port	uint	0-2	(see below)
8				x	Location-Path	string	0-255	(none)
11	x	x	-	x	Uri-Path	string	0-255	(none)
12					Content-Format	uint	0-2	(none)
14		x	-		Max-Age	uint	0-4	60
15	x	x	-	x	Uri-Query	string	0-255	(none)
16					Accept	uint	0-2	(none)
20				x	Location-Query	string	0-255	(none)
35	x	x	-		Proxy-Uri	string	1-1034	(none)
39	x	x	-		Proxy-Scheme	string	1-255	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

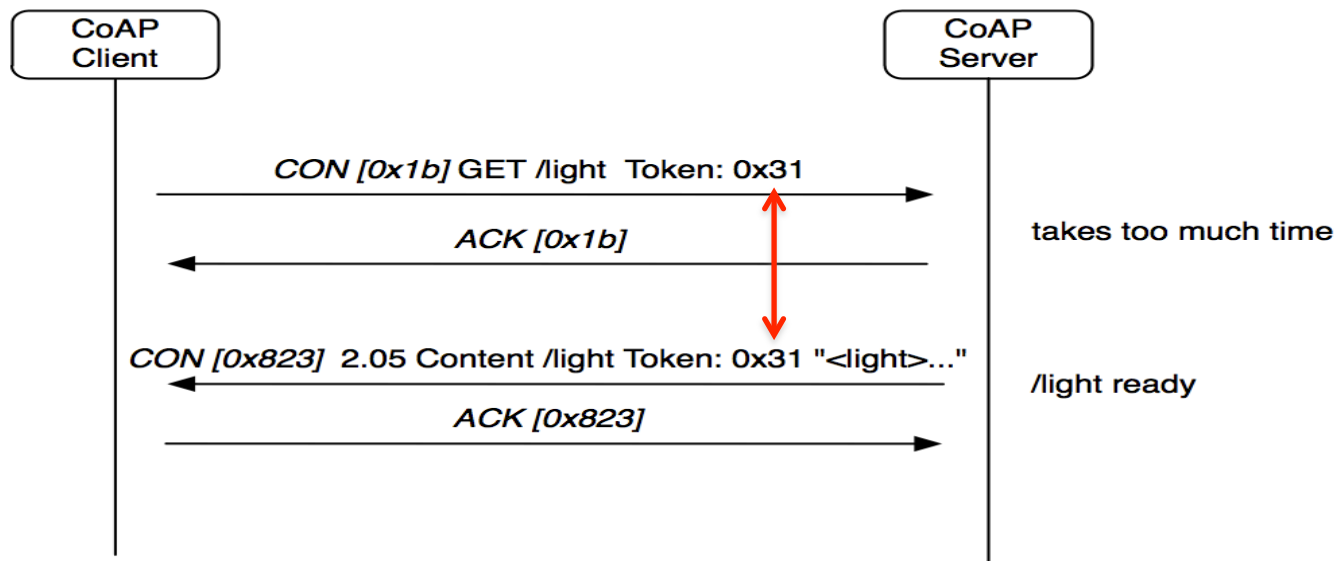
Request Example



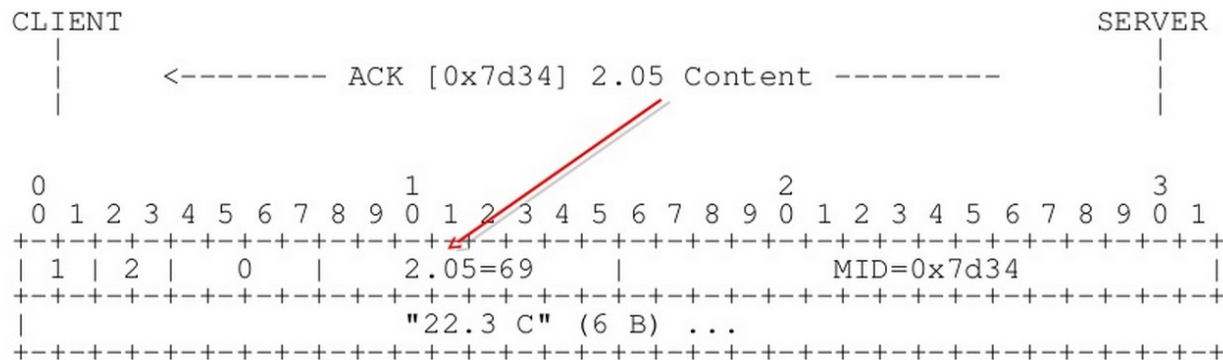
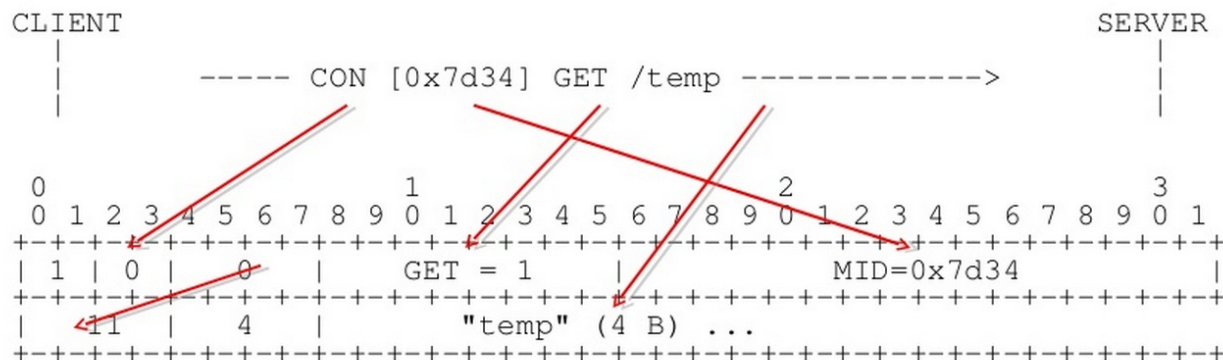
Dealing with Packet Loss



Separate Response



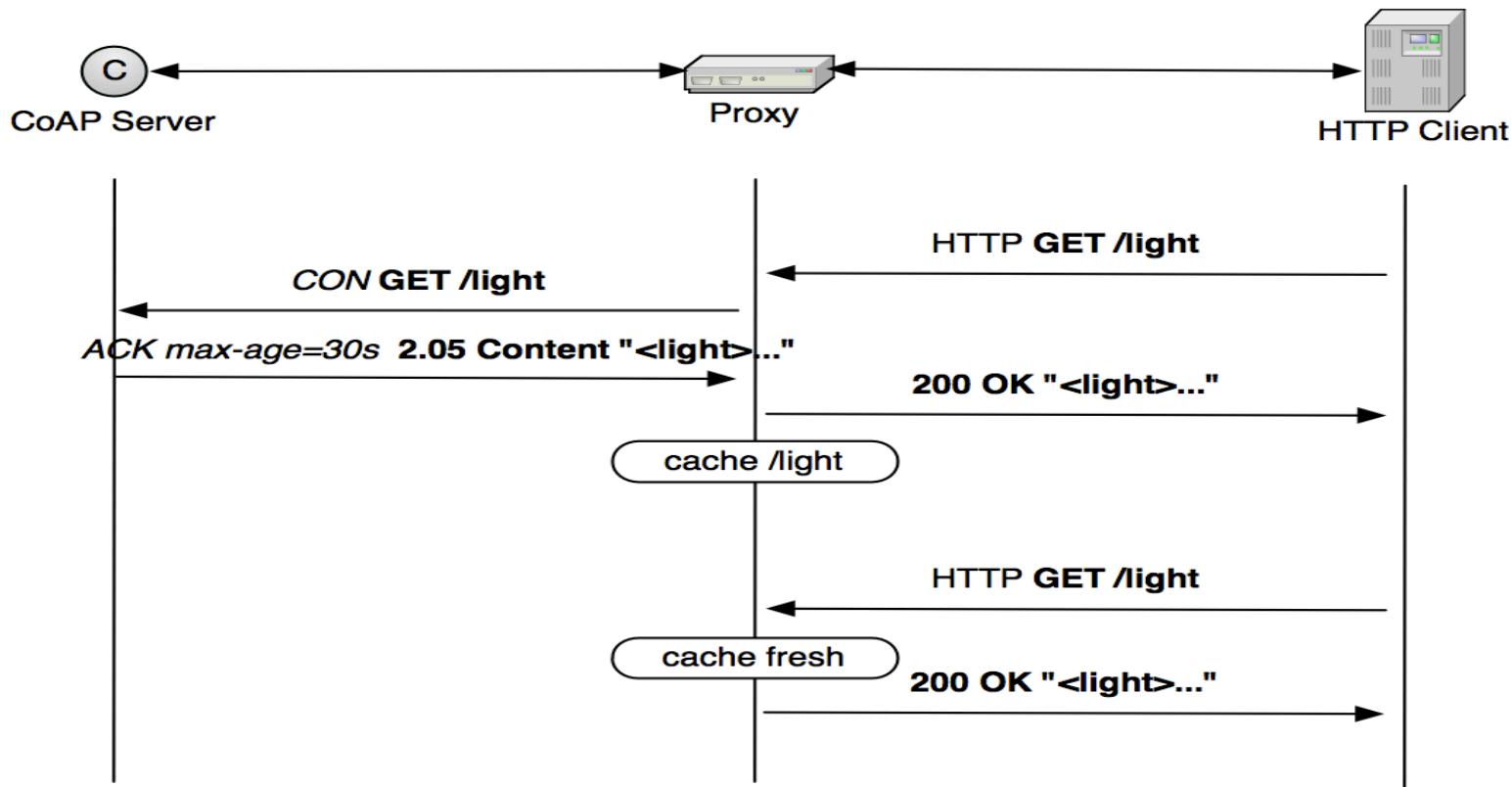
Bits and bytes...



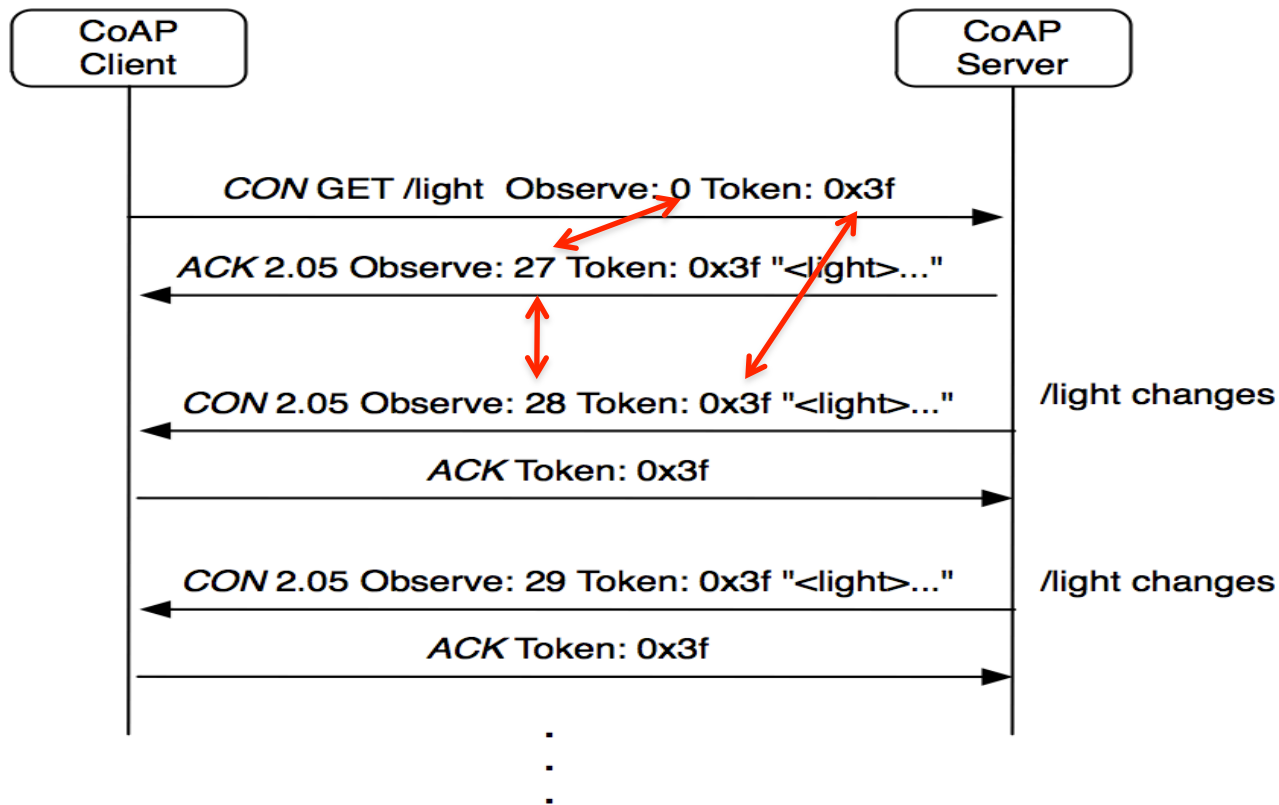
Caching

- CoAP includes a simple caching model
 - Cacheability determined by response code
 - An option number mask determines if it is a cache key
- Freshness model
 - Max-Age option indicates cache lifetime
- Validation model
 - Validity checked using the Etag Option
- A proxy often supports caching
 - Usually on behalf of a constrained node,
 - a sleeping node,
 - or to reduce network load

Proxying and caching

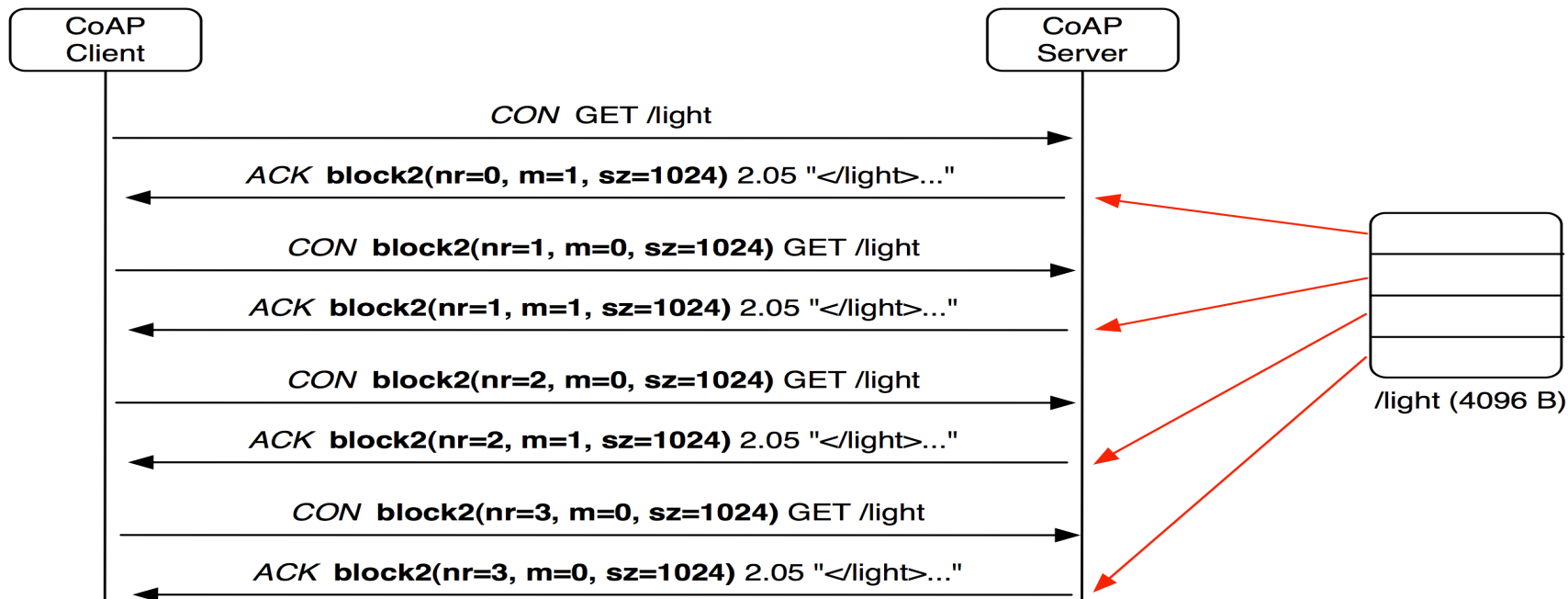


Observation



See draft-ietf-core-observe

Block transfer



Getting Started with CoAP

- There are many open source implementations available
 - [mbed](#) includes CoAP support
 - 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
 - ARM 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!