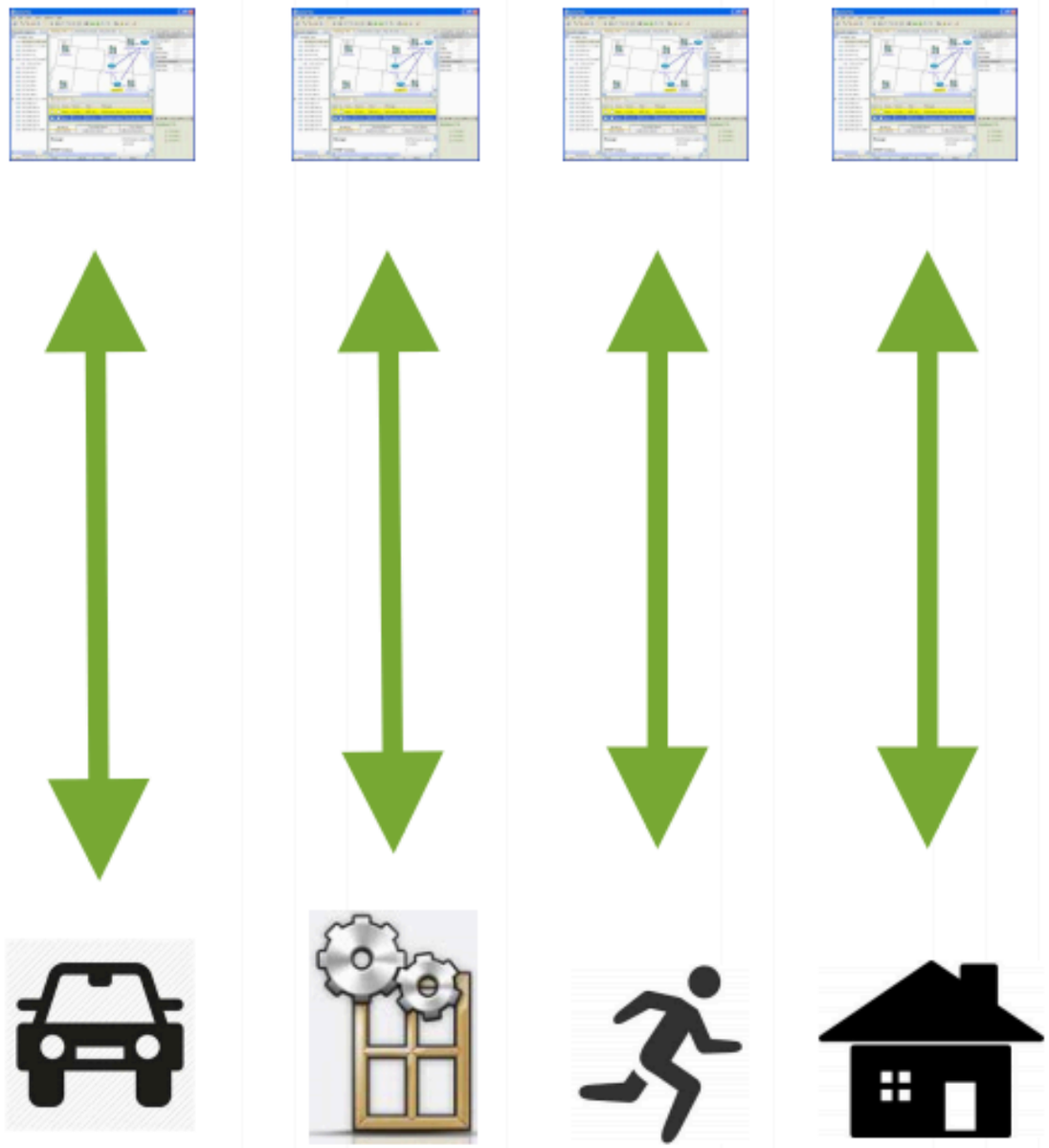
Lecture #9
CoAP - Constrained Application Protocol

Android Things 2020


Internet
of people

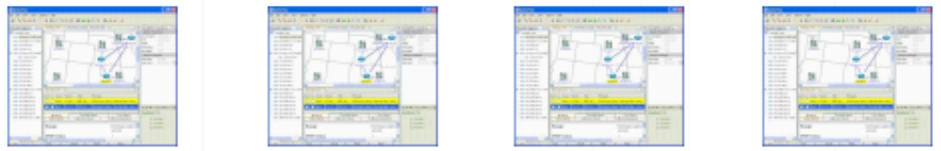

Internet
of things

M2M



Big Data

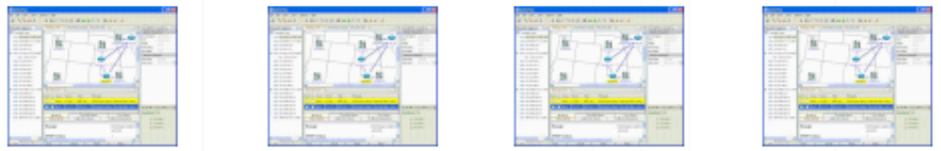
Internet of Things



Little Data

Big Data

Internet of Things



Little Data

Services

|

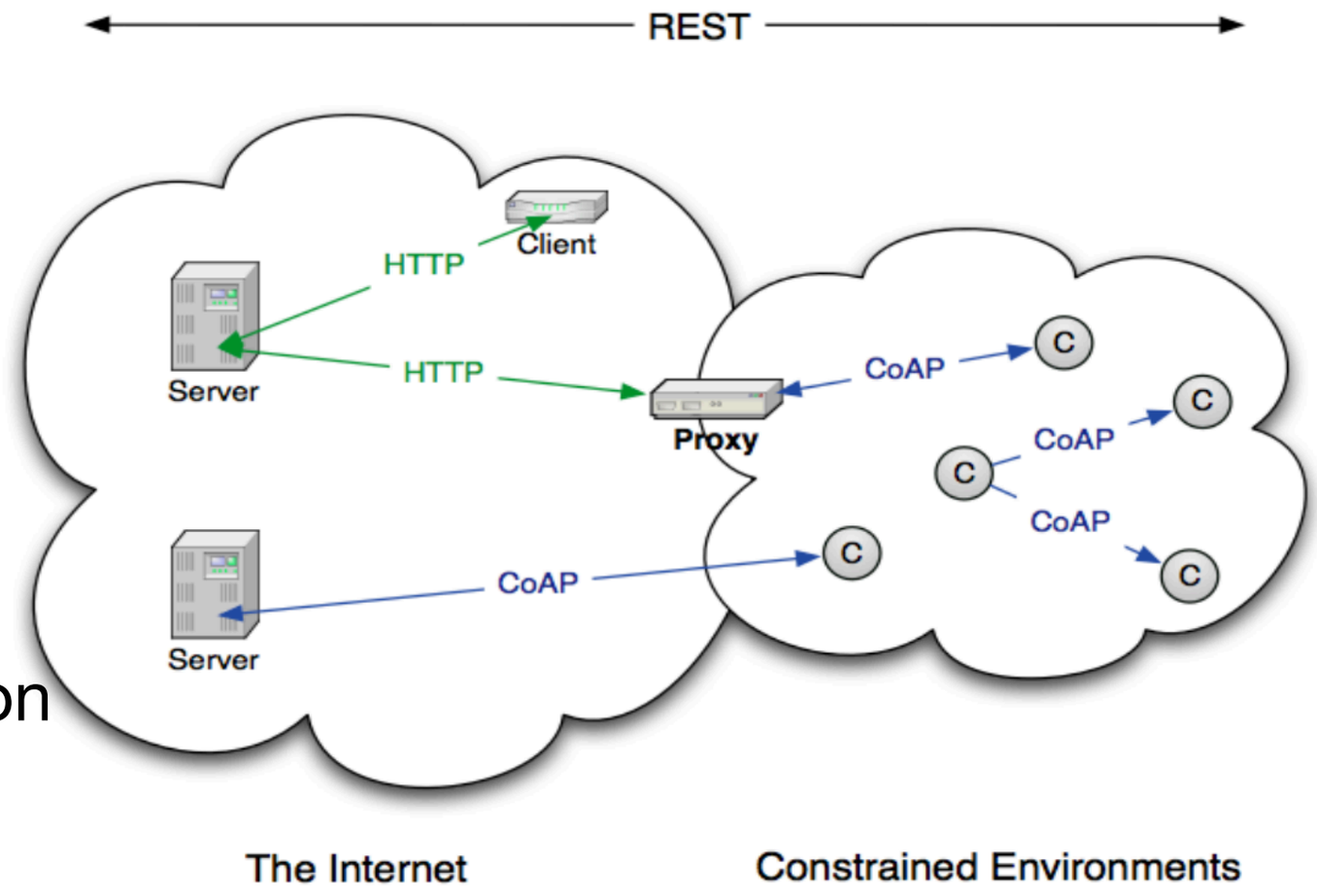
The Web

|

Things

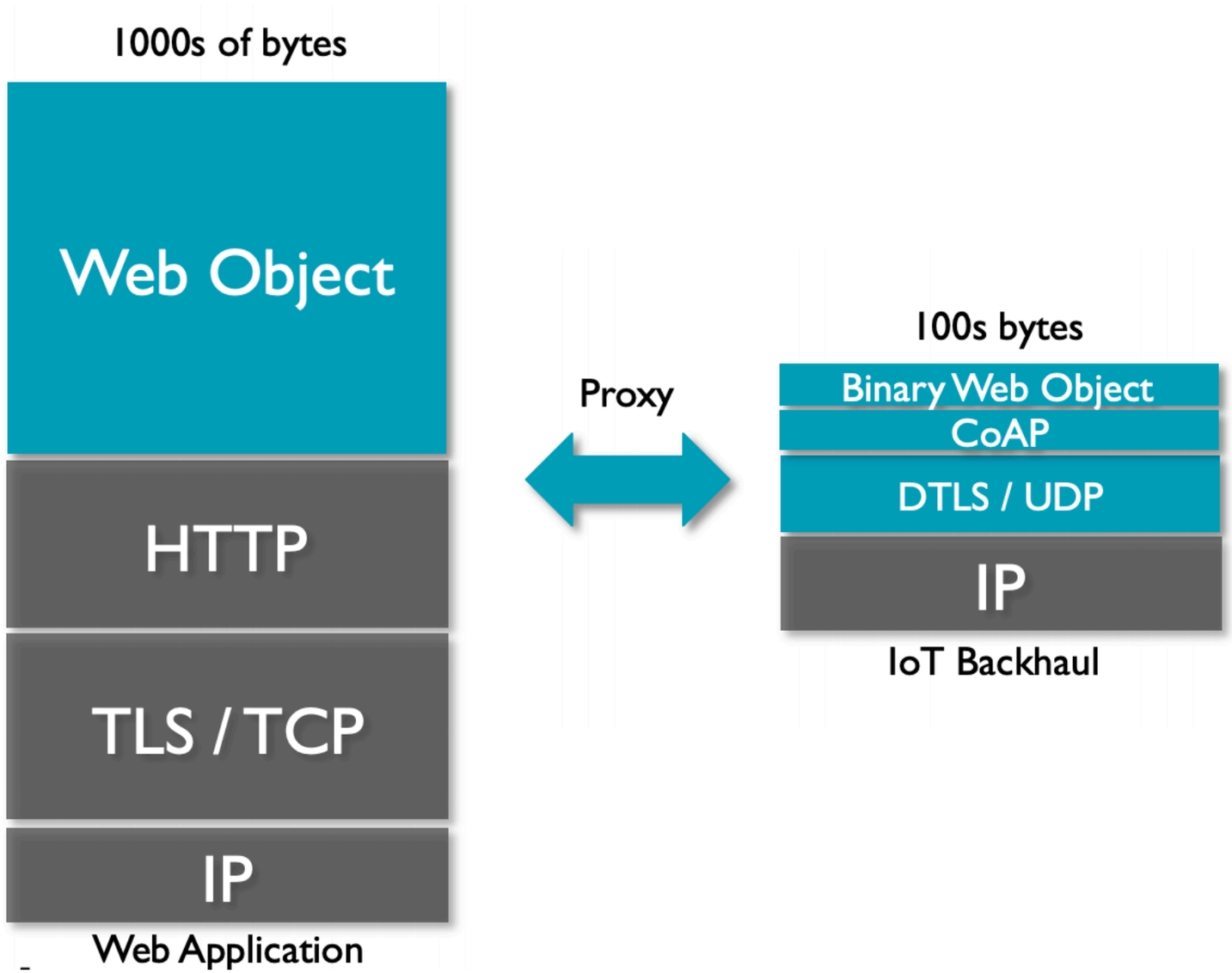
CoAP

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

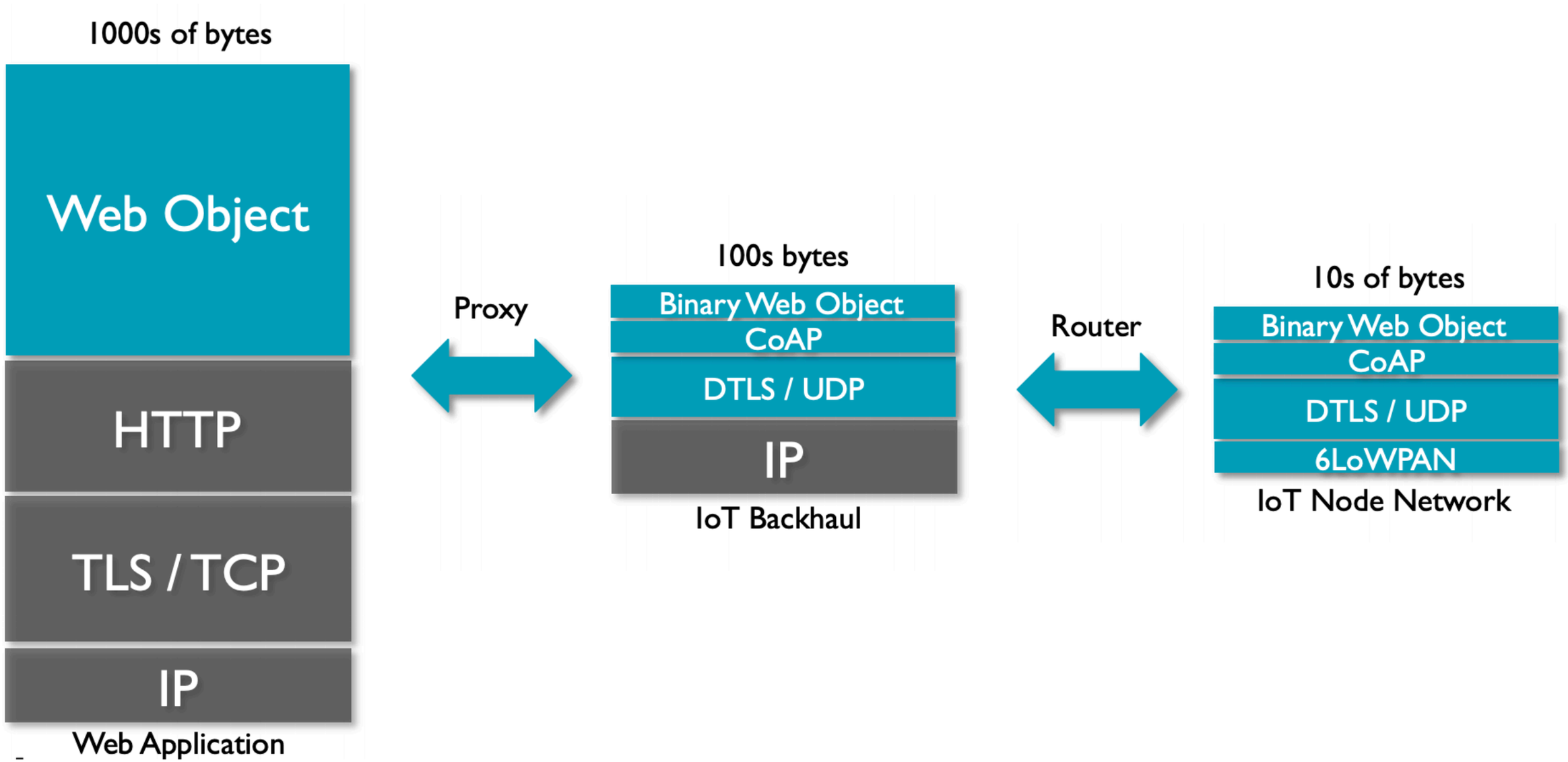


Transitions from Web to IoT

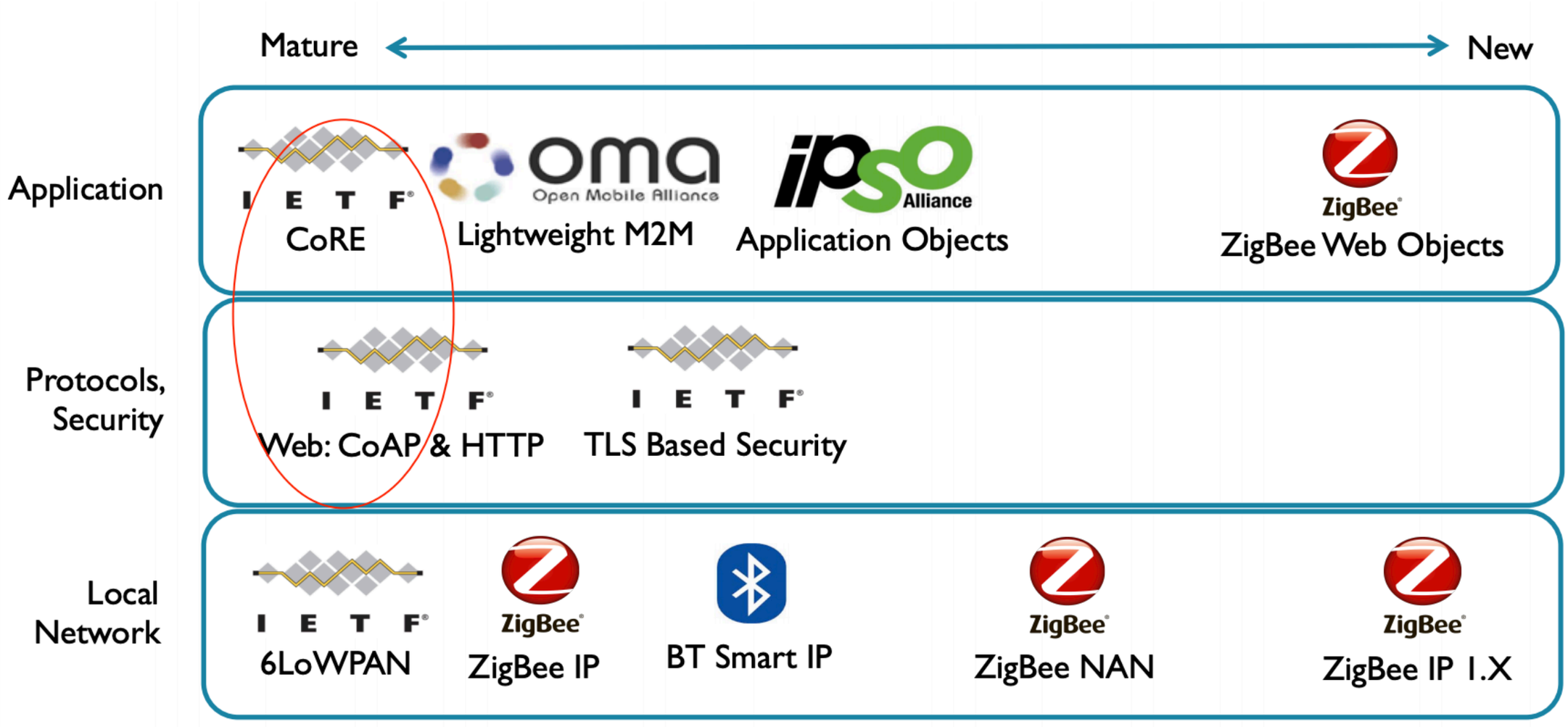
Transitions from Web to IoT



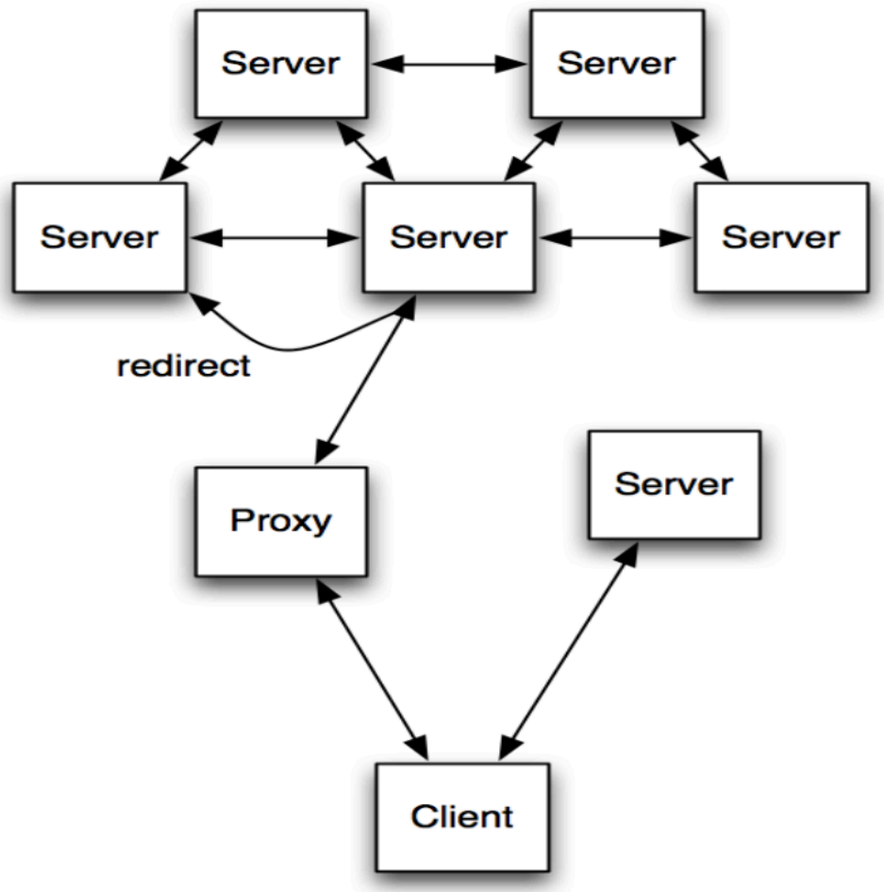
Transitions from Web to IoT



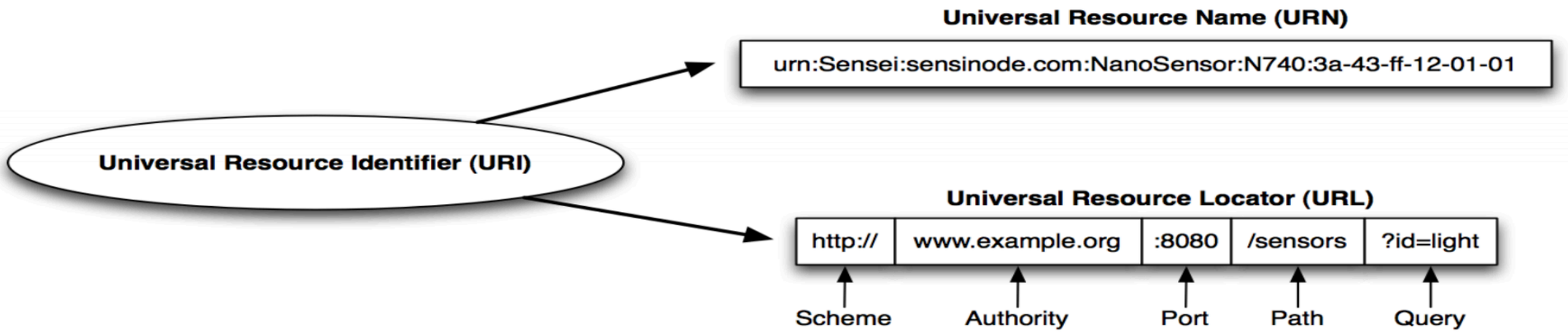
Community


Mature
New
Application
Protocols, Security
Local Network
IETF
CoRE
oma
Open Mobile Alliance
Lightweight M2M
ipso
Alliance
Application Objects
ZigBee
ZigBee Web Objects
Web: CoAP & HTTP
TLS Based Security
6LoWPAN
ZigBee IP
BT Smart IP
ZigBee NAN
ZigBee IP 1.X

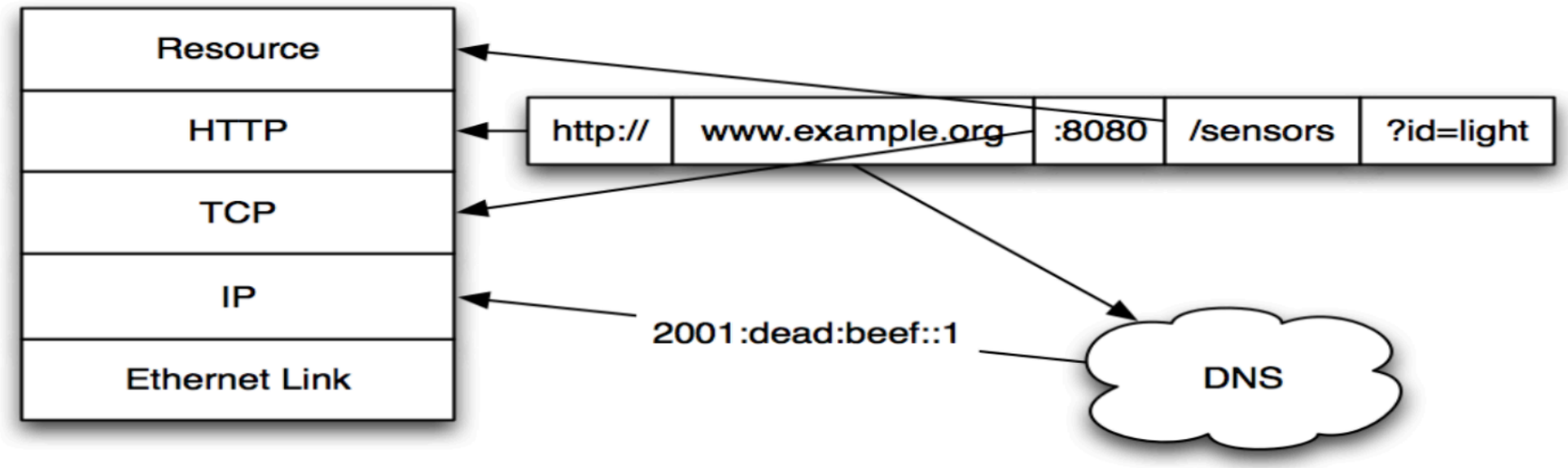
Web Architecture



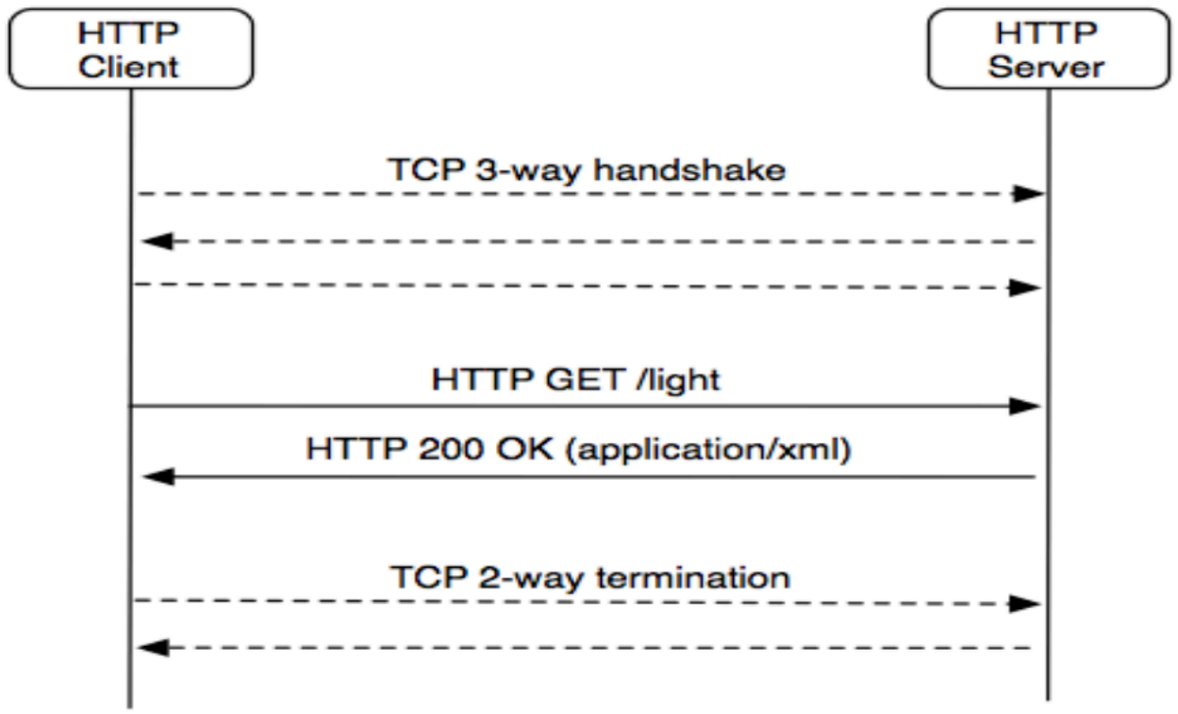
Naming


Universal Resource Name (URN)
urn:Sensei:sensinode.com:NanoSensor:N740:3a-43-ff-12-01-01
Universal Resource Identifier (URI)
Universal Resource Locator (URL)
http://
www.example.org
:8080
/sensors
?id=light
Scheme
Authority
Port
Path
Query

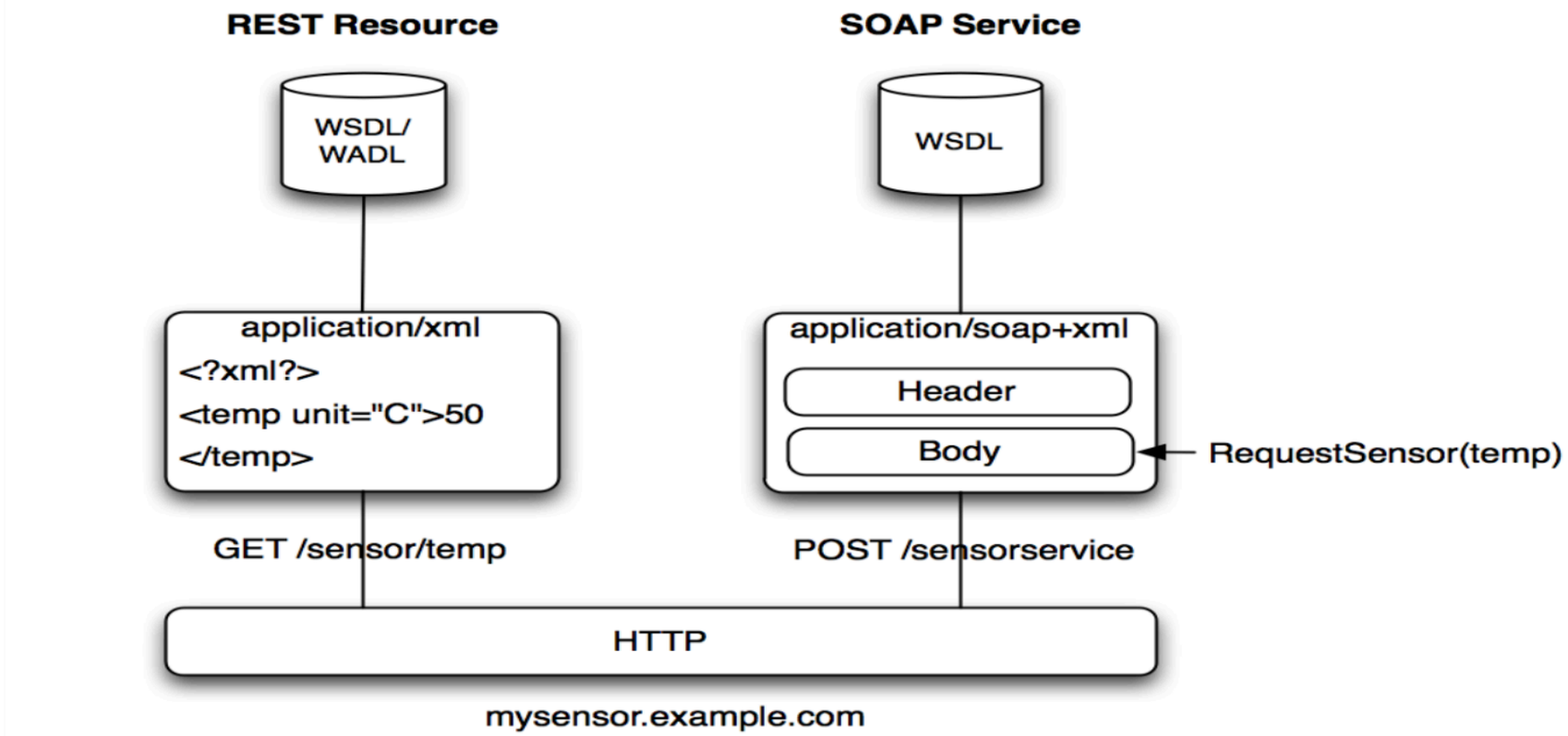
Resolution


Resource
HTTP
TCP
IP
Ethernet Link
http://
www.example.org
:8080
/sensors
?id=light
2001:dead:beef::1
DNS

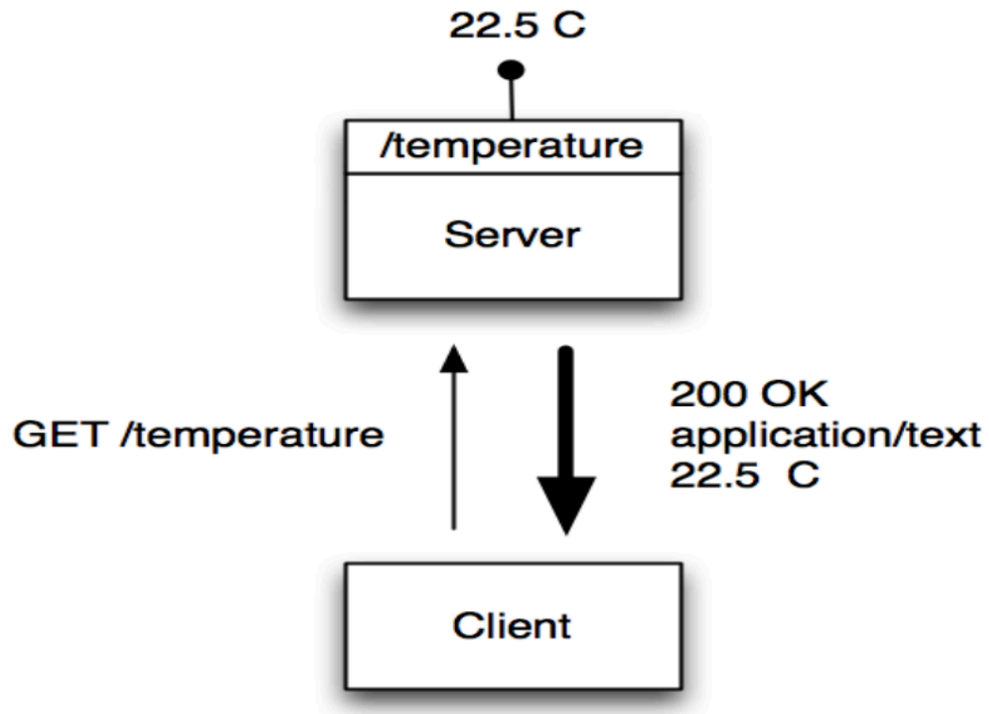
Traditional HTTP



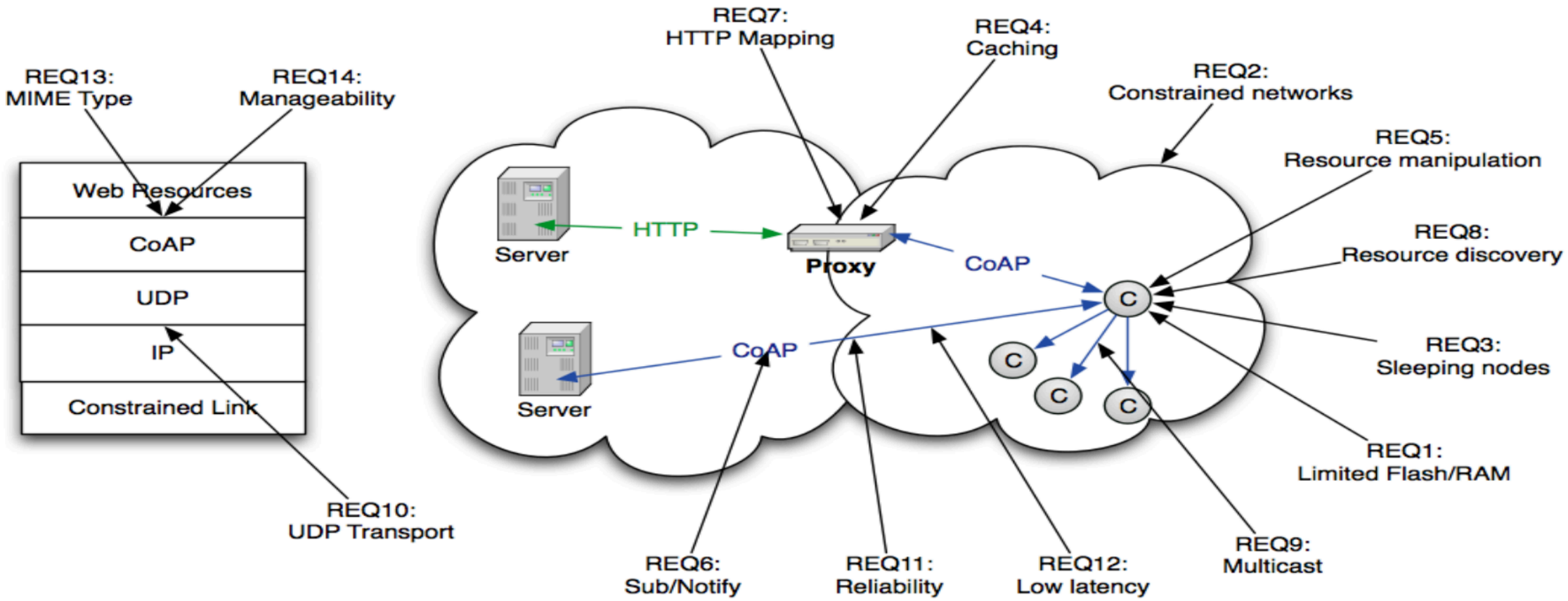
Web Paradigms



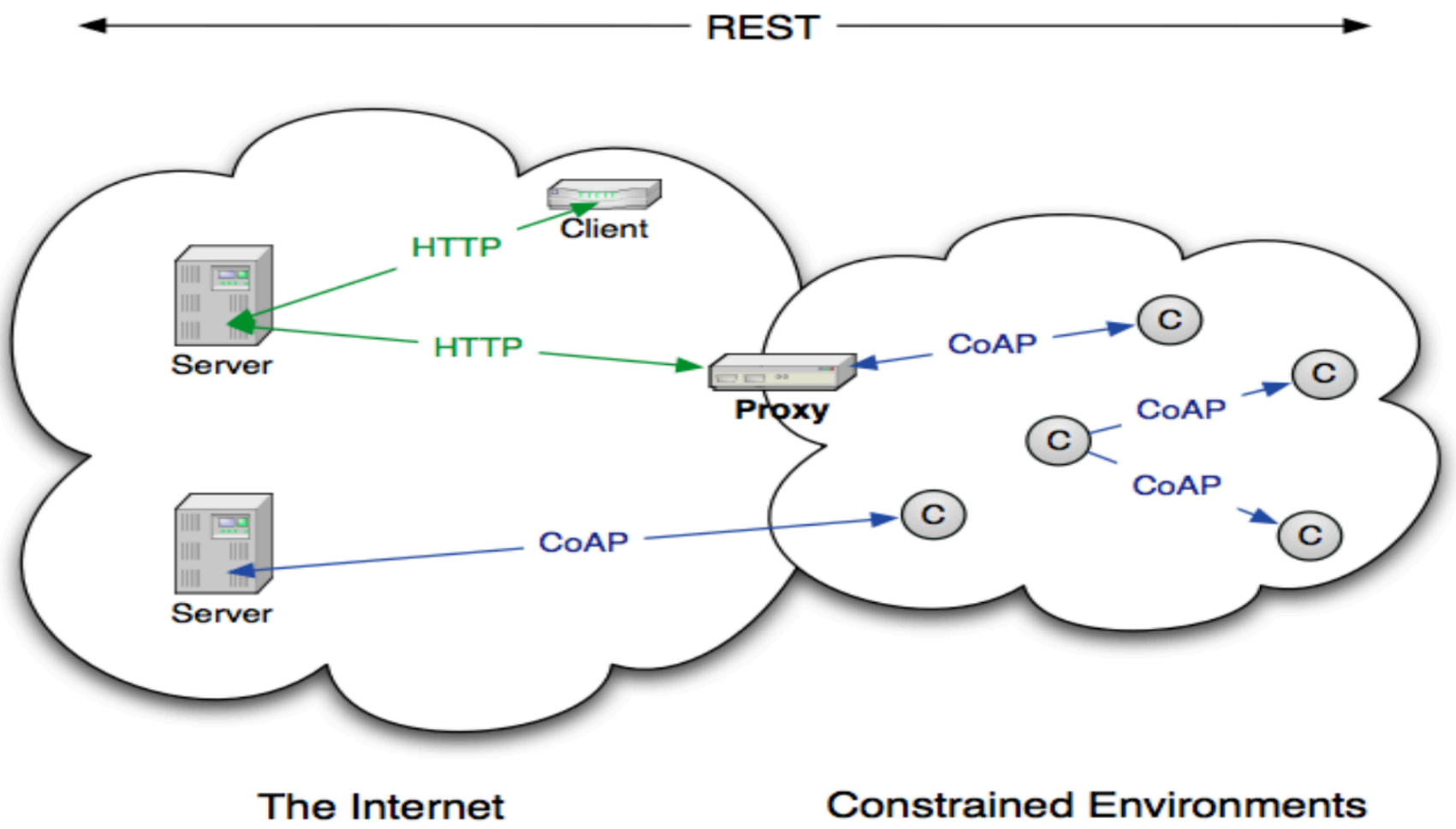
REST Request



CoAP

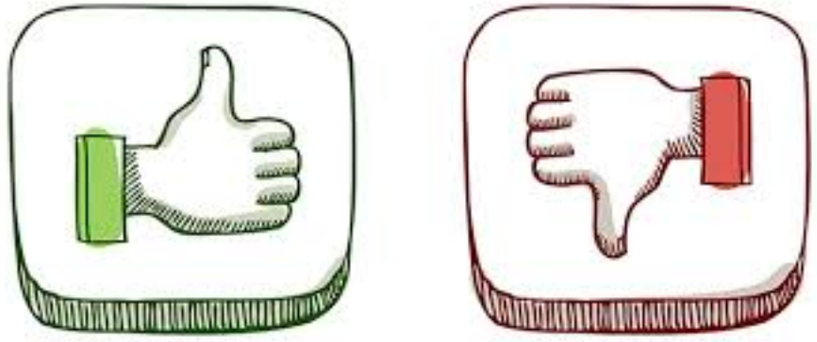

REQ13:
MIME Type
REQ14:
Manageability
Web Resources
CoAP
UDP
IP
Constrained Link
REQ10:
UDP Transport
REQ7:
HTTP Mapping
REQ4:
Caching
REQ2:
Constrained networks
REQ5:
Resource manipulation
REQ8:
Resource discovery
REQ3:
Sleeping nodes
REQ1:
Limited Flash/RAM
REQ9:
Multicast
REQ12:
Low latency
REQ11:
Reliability
REQ6:
Sub/Notify
Server
HTTP
Proxy
CoAP
Server
CoAP
C
C
C
C

Architecture



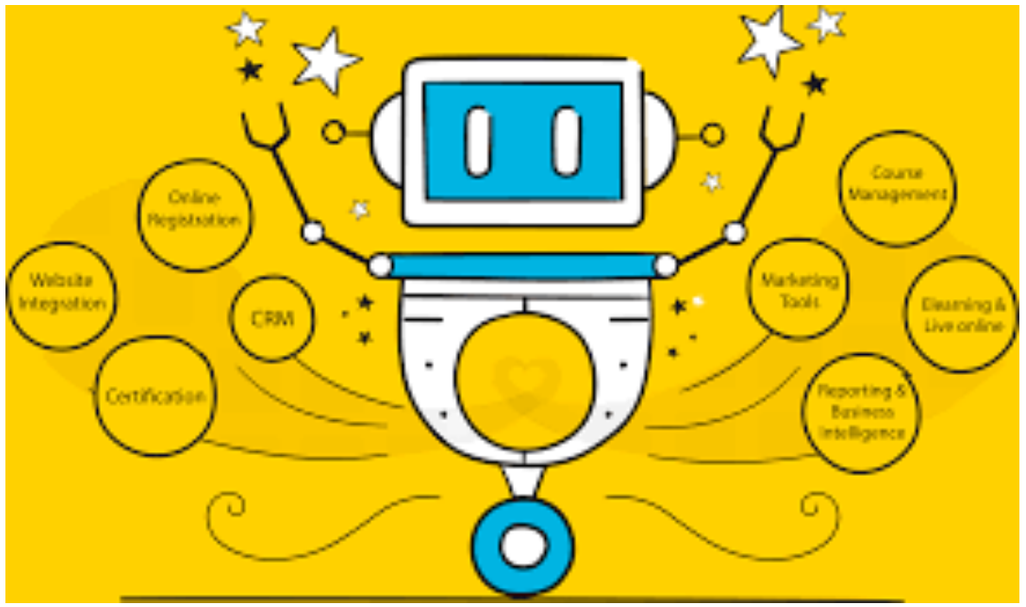
Pro/Cons

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



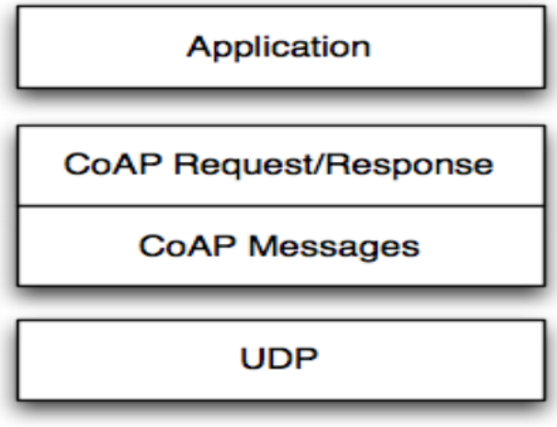
Features

- 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

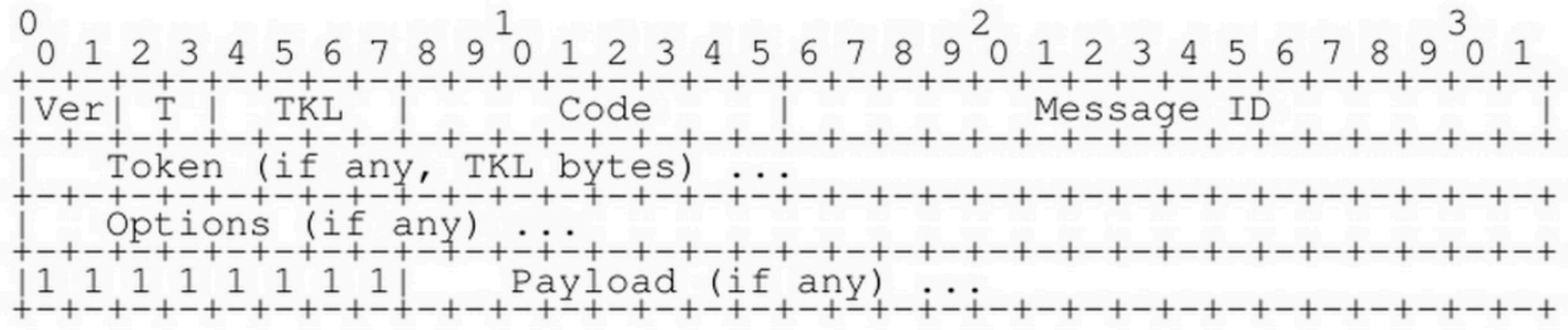


Transactional Model

- 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.)



Header



Ver - Version (1)

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

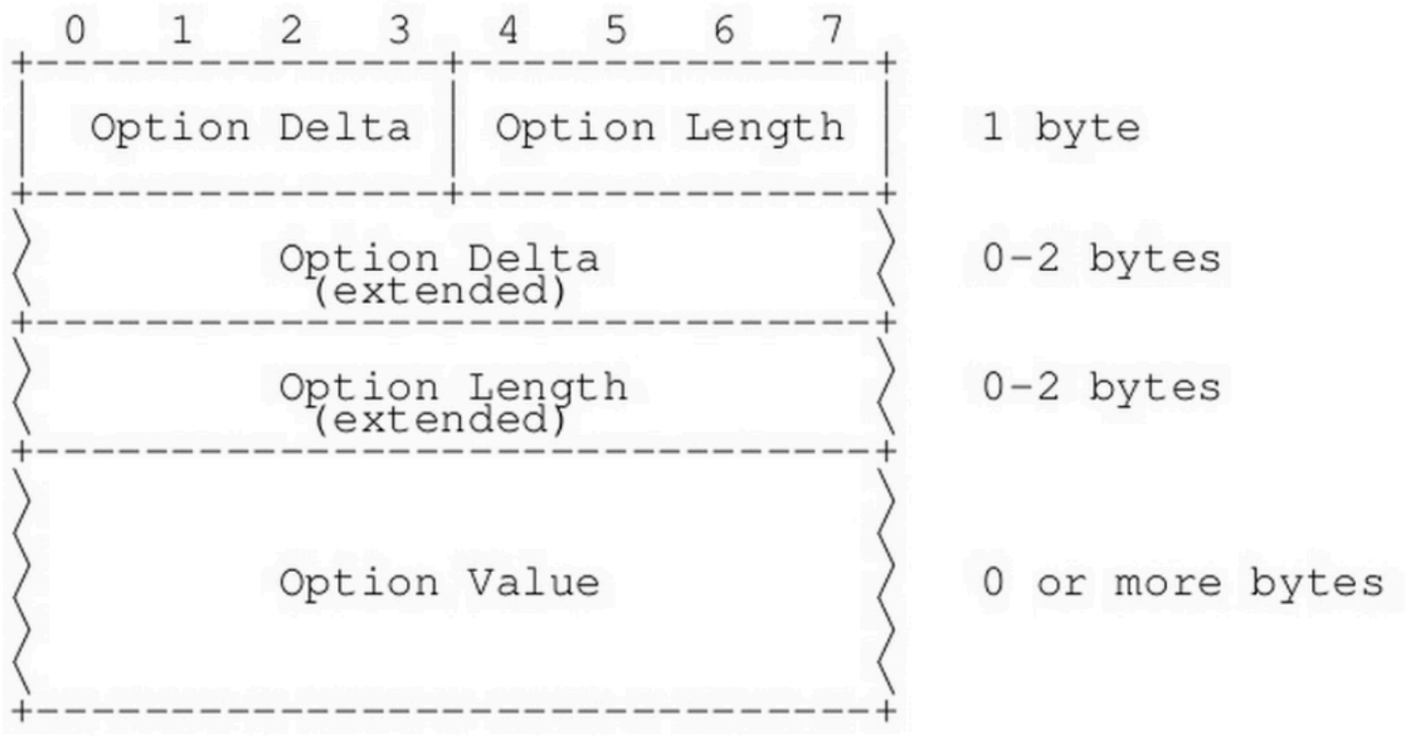
TKL- Token Length, if any, the number of Token bytes after this header

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

Message ID - 16-bit identifier for matching responses

Token - Optional response matching token

Options Field



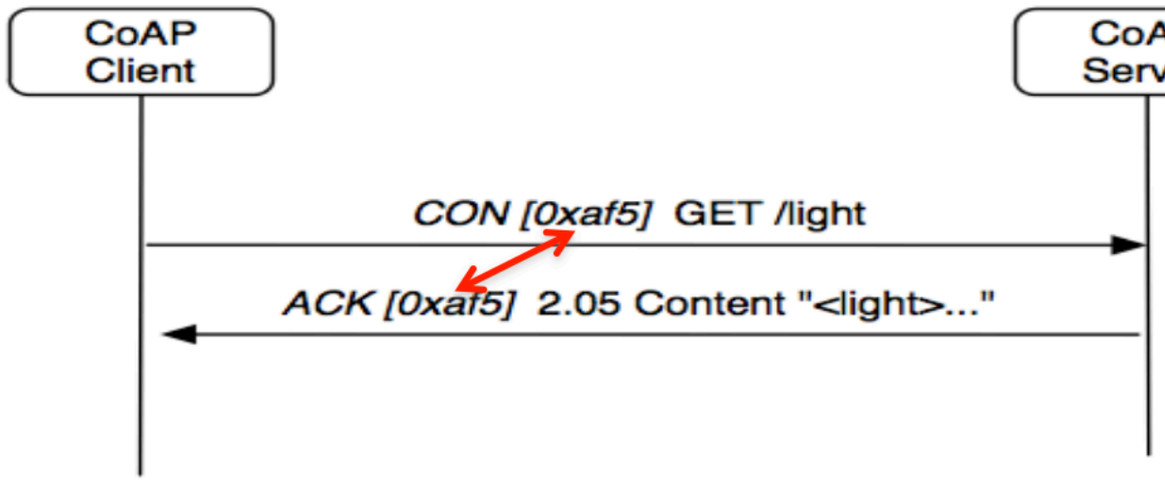
Option Delta - Difference between this option type and the previous
Length - Length of the option value
Value - The value of Length bytes immediately follows Length

Base Specification

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

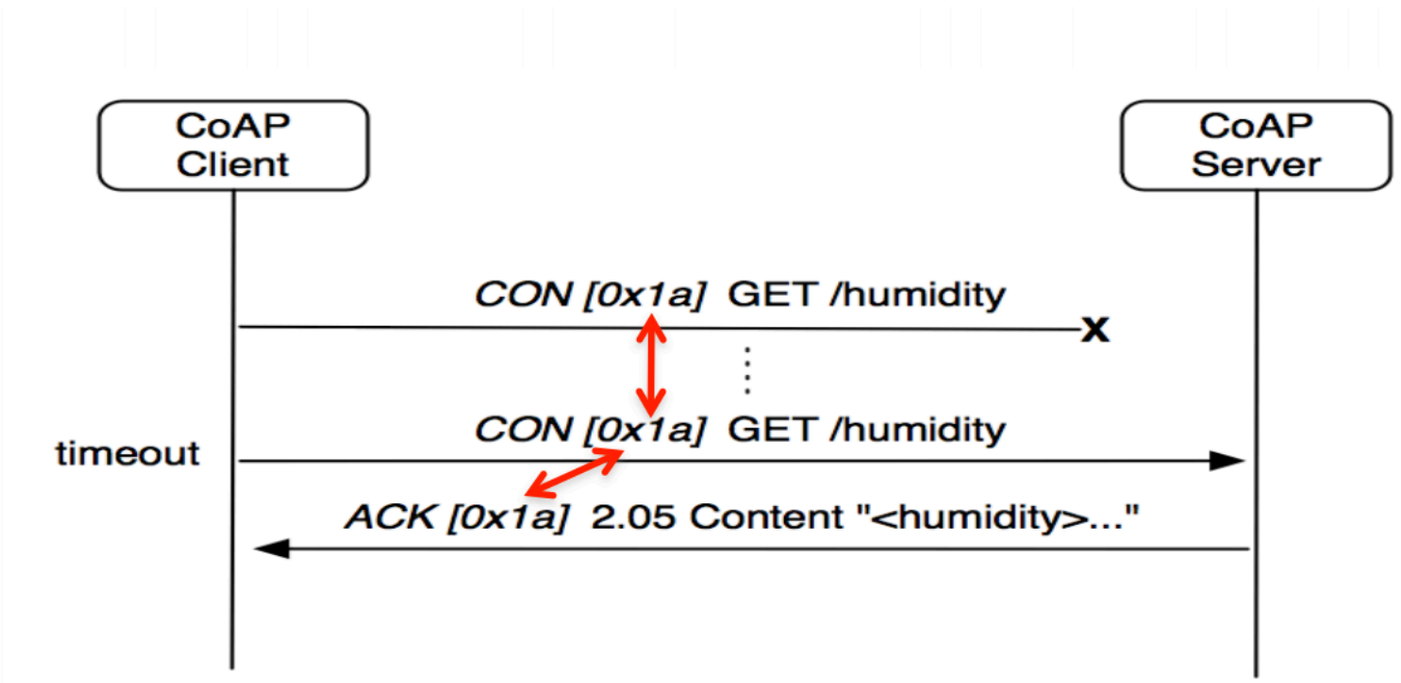
Simple Request



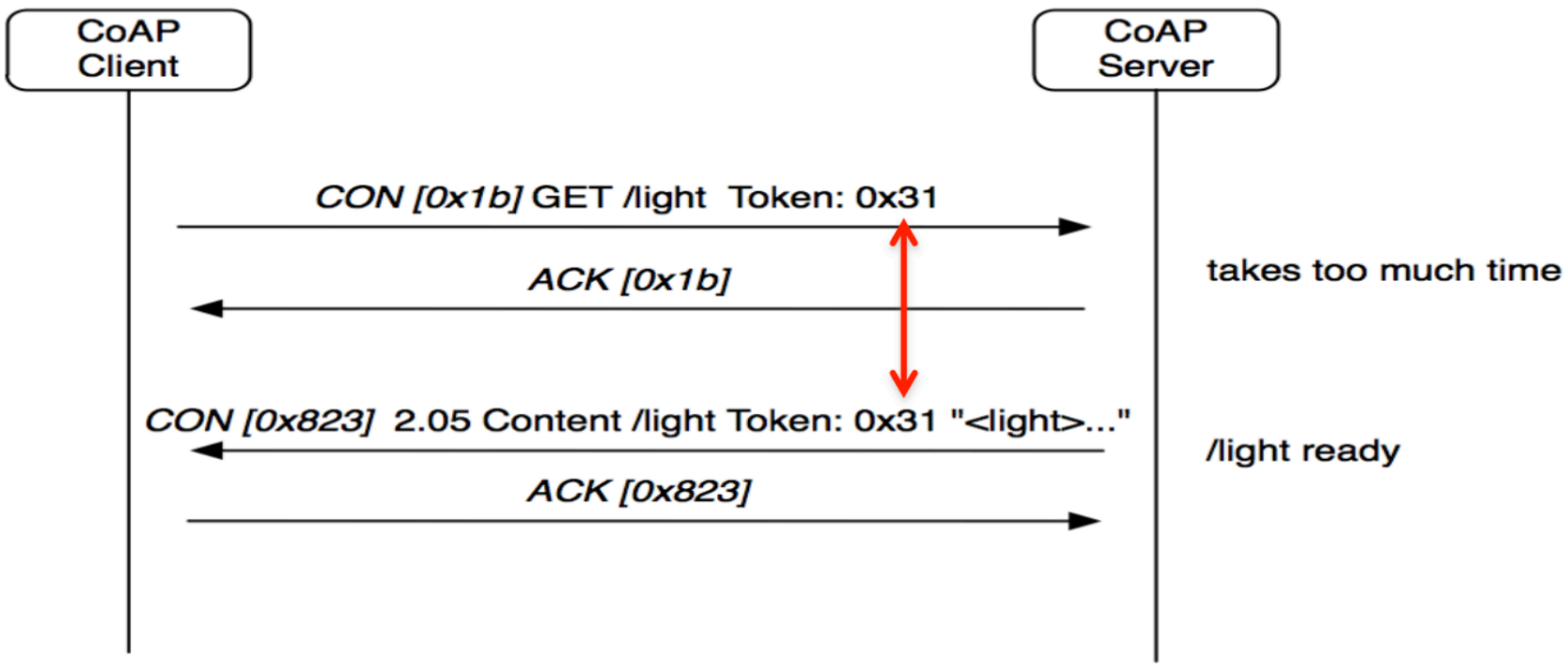
Confirmable Request

Piggy-backed Response

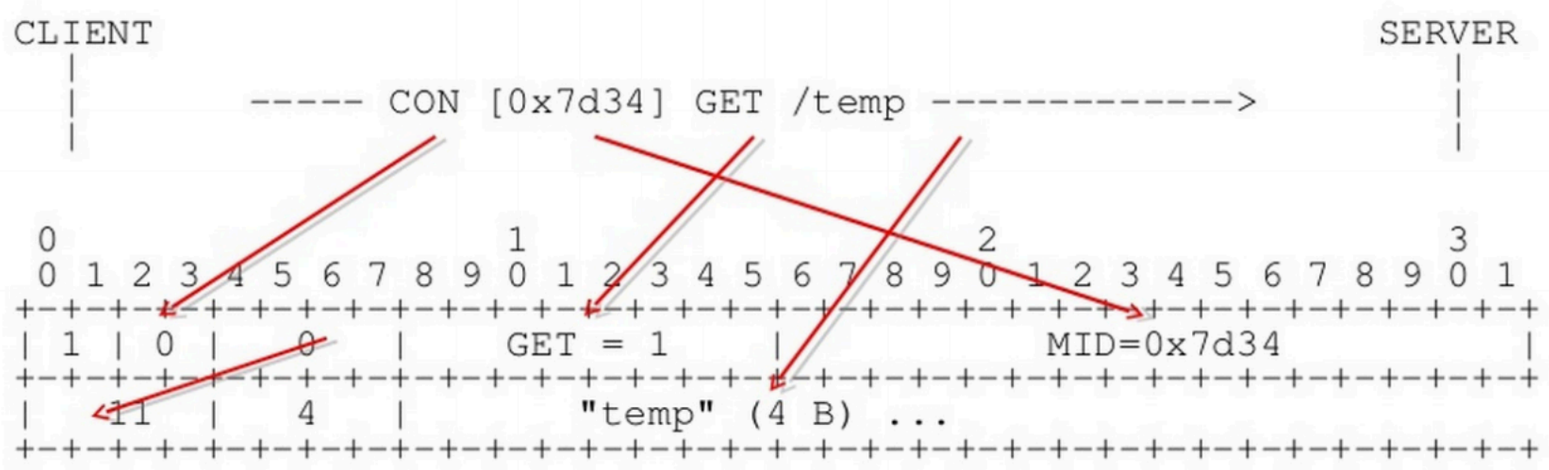
Data Loss

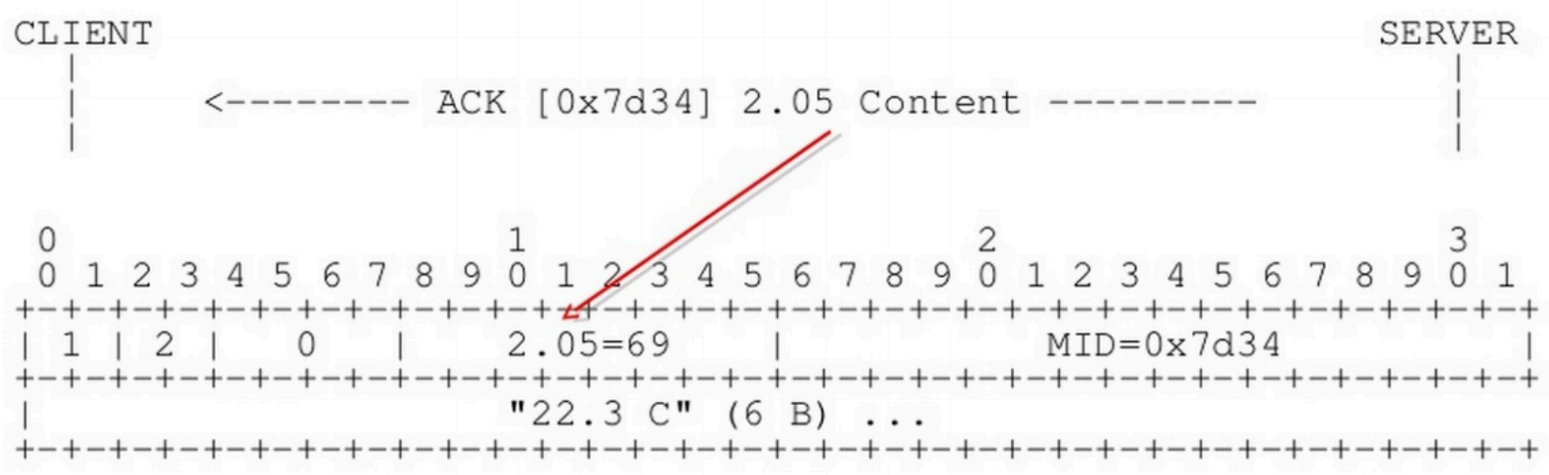

CoAP Client
CoAP Server
CON [0x1a] GET /humidity
X
CON [0x1a] GET /humidity
timeout
ACK [0x1a] 2.05 Content "<humidity>..."

Separate Response



Bits & Bytes

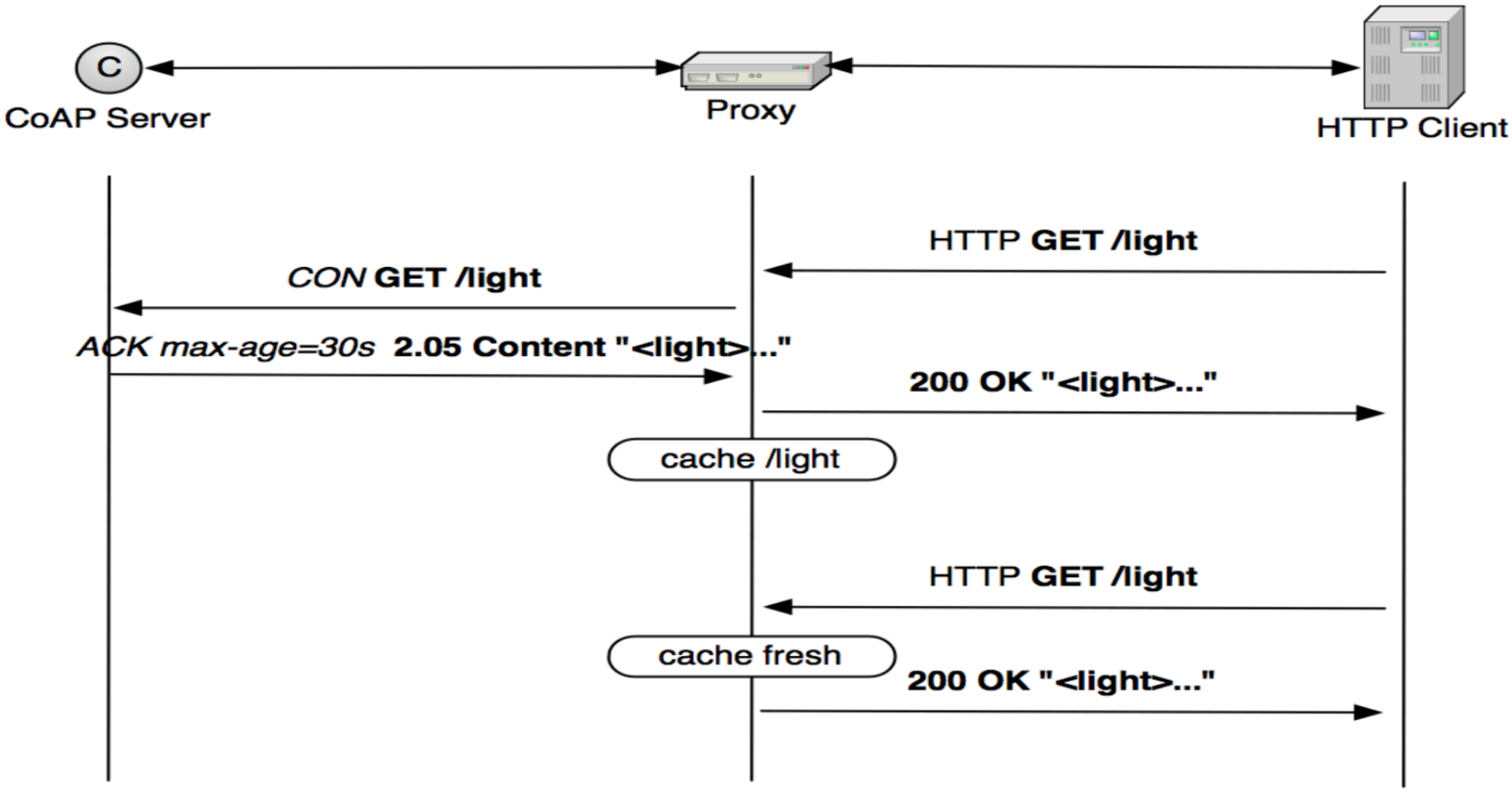

CLIENT
SERVER
----- CON [0x7d34] GET /temp ------------>
0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
| 1 | 0 | 0 | GET = 1 | MID=0x7d34 |
| 11 | 4 | "temp" (4 B) ... |


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) ... |

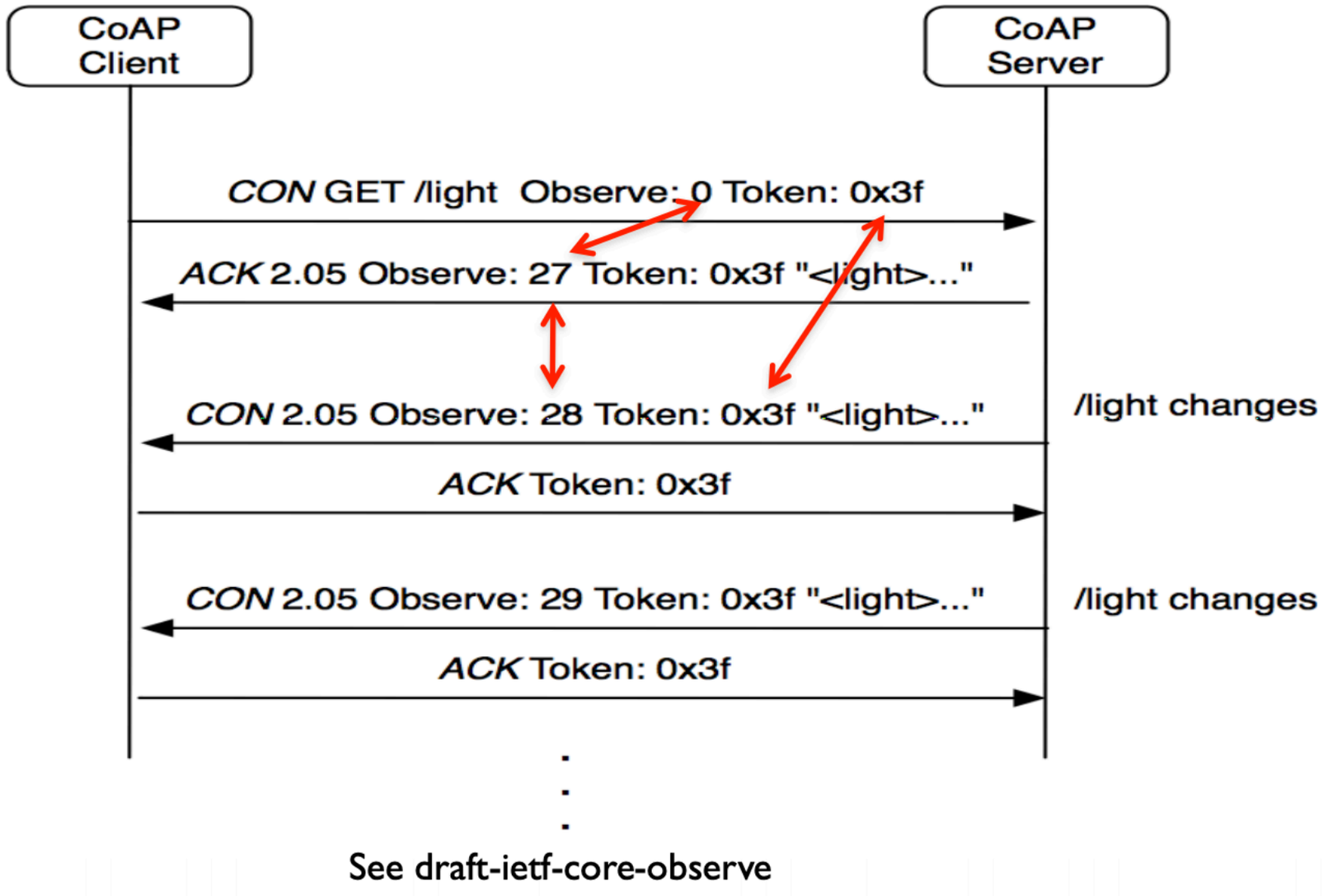
Caching

- CoAP includes a simple caching model
  - 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.

Proxy

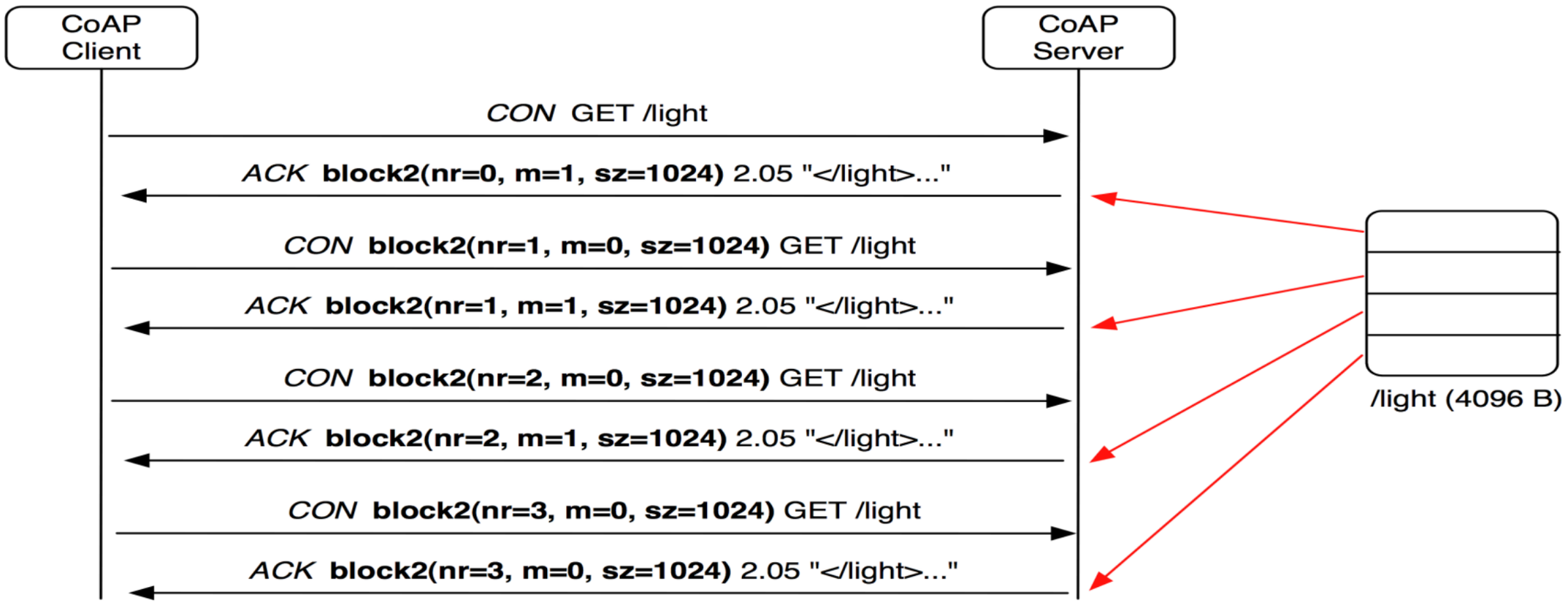

C
CoAP Server
Proxy
HTTP Client
HTTP GET /light
CON GET /light
ACK max-age=30s 2.05 Content "<light>..."
200 OK "<light>..."
cache /light
HTTP GET /light
cache fresh
200 OK "<light>..."

Subscription



See draft-ietf-core-observe

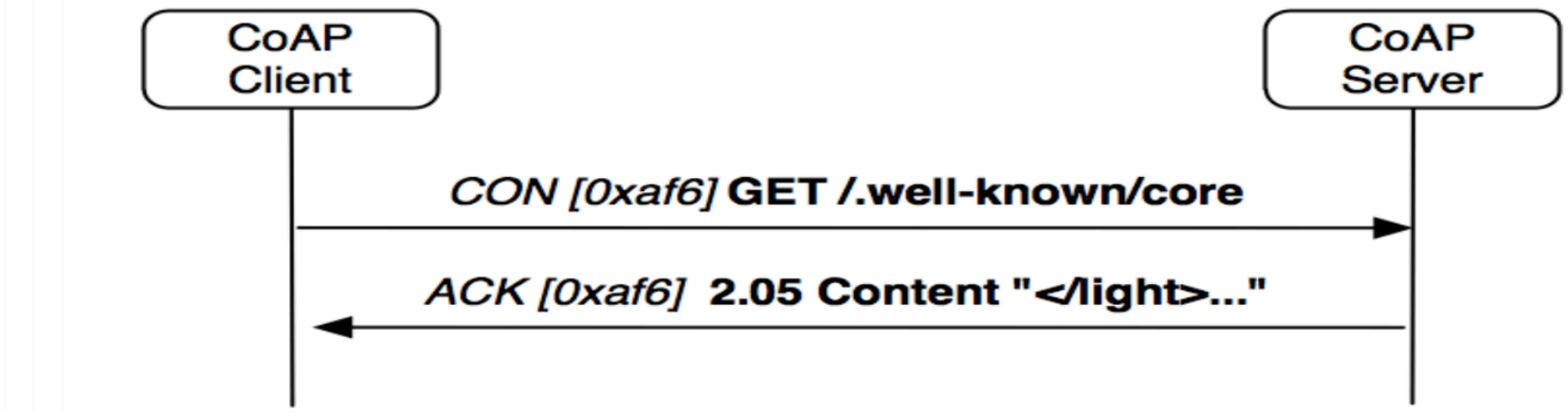
Block Transfer



Community & Open Source

- 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!

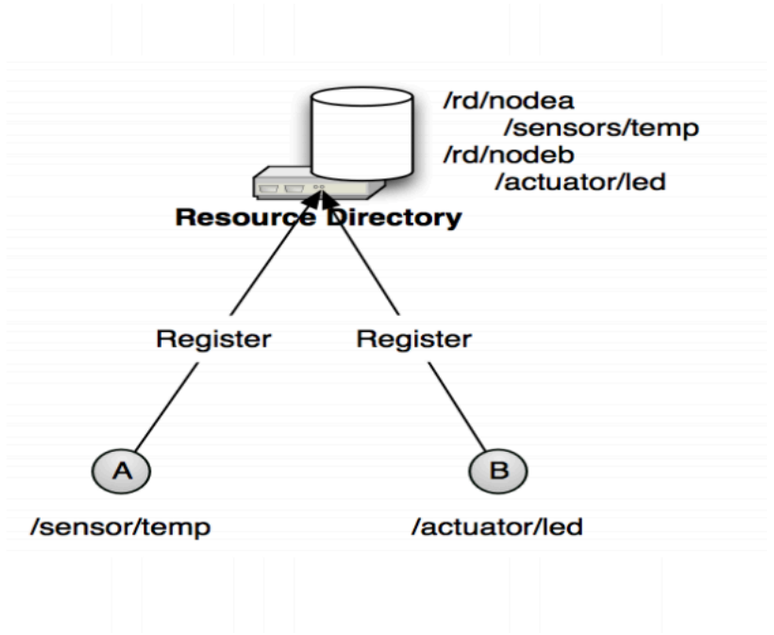
Resource Discovery




```
</dev/bat>;obs;rt="ipso:dev-bat";ct="0",
</dev/mdl>;rt="ipso:dev-mdl";ct="0",
</dev/mfg>;rt="ipso:dev-mfg";ct="0”,
</pwr/0/rel>;obs;rt="ipso:pwr-rel";ct="0",
</pwr/0/w>;obs;rt="ipso:pwr-w";ct="0",
</sen/temp>;obs;rt="ucum:Cel";ct="0"
```

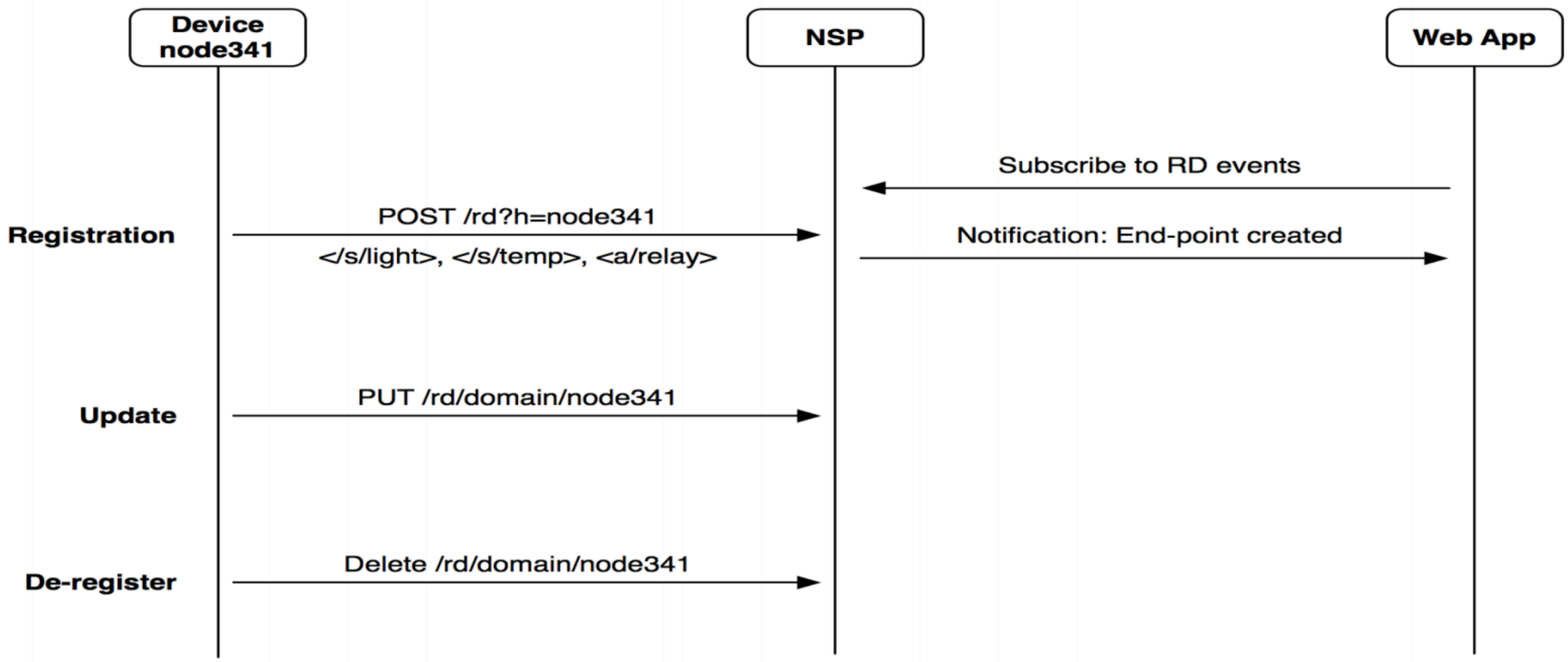
Resource Directory

- Link Format only defines
  - The link format
  - Peer-to-peer discovery
- A directory approach is also useful
  - Supports sleeping nodes
    - No multicast traffic, longer battery life
  - Remote lookup, hierarchical and federated distribution
- The CoRE Link Format can be used to build Resource Directories
  - Nodes POST (register) their link-format to an RD
  - Nodes PUT (refresh) to the RD periodically
  - Nodes may DELETE (remove) their RD entry
  - Nodes may GET (lookup) the RD or resource of other nodes



DEMO

Resource Directory



Lecture outcomes

- CoAP Protocol.
- Practice using a sample.