sensinode

May 19th, 2013

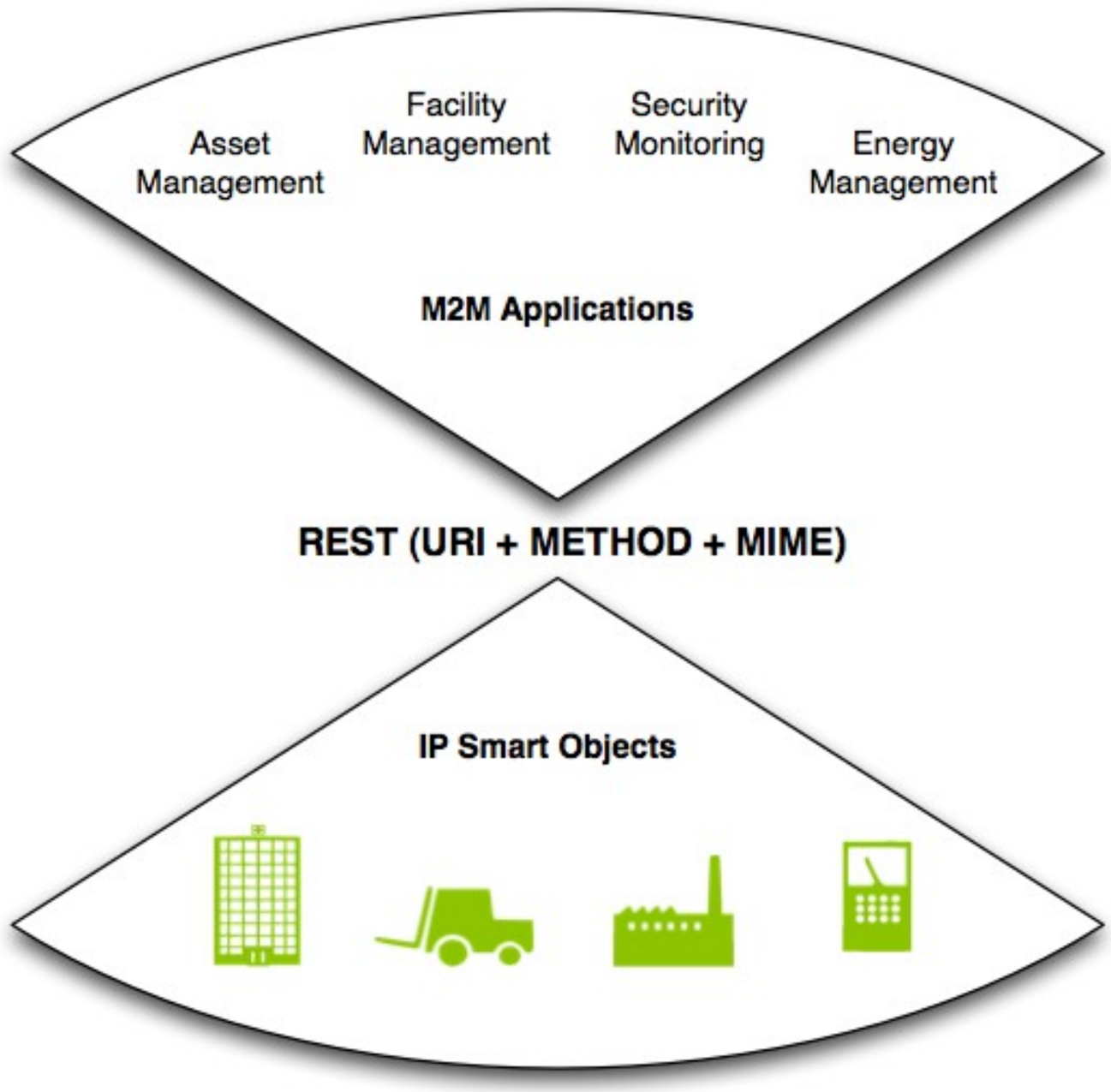
CoAP: The Internet of Things Protocol

Zach Shelby, Chief Nerd

Tutorial Overview

- The Web of Things
- Example Applications
- The Web & REST?
- Constrained Application Protocol (CoAP)
  - ✓ Base CoAP Specification
  - ✓ Observation
  - ✓ Block Transfer
  - ✓ Getting Started with CoAP
- Discovery and Semantics
- OMA Lightweight M2M

The Web of Things



Key IoT Standardization

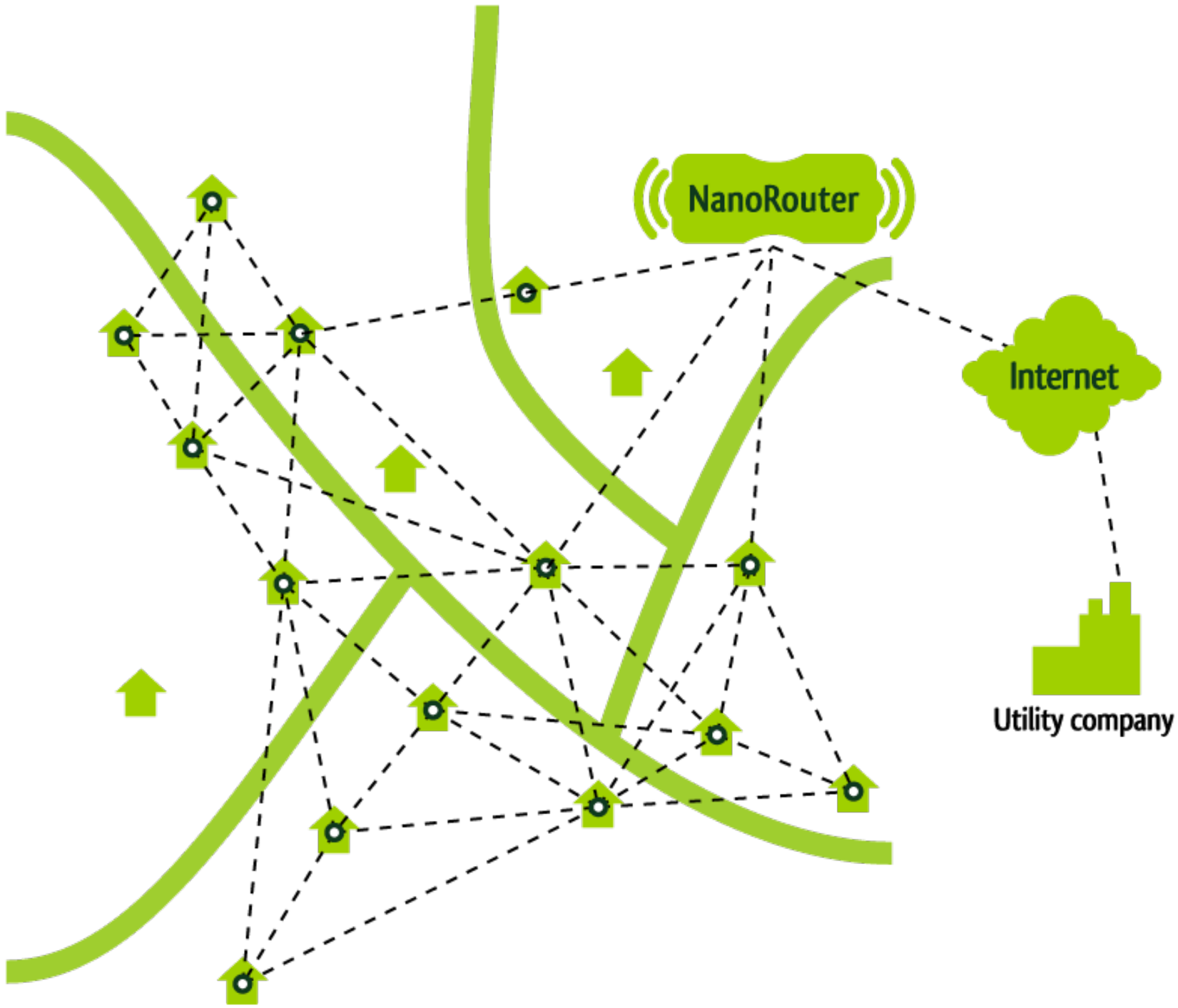
- **IETF**
  - ✓ 6LoWPAN Working Group (IPv6 anywhere)
  - ✓ ROLL (Routing Over Low-power Lossy Networks) WG
  - ✓ CoRE WG (REST for IoT, CoAP, Resource Directory etc.)
  - ✓ TLS WG (DTLS)
- **OMA**
  - ✓ Lightweight M2M Enabler Standard (CoAP/DTLS based)
  - ✓ Device Management 2.0 Enabler Standard (HTTP/TLS based)
- **ETSI / OneM2M**
  - ✓ Ongoing work on M2M system standardization (CoAP, HTTP binding)
- **W3C**
  - ✓ Efficient XML Interchange (EXI) standardization
- **ZigBee IP**
  - ✓ An open-standard 6LoWPAN stack for e.g. Smart Energy 2.0

sensinode

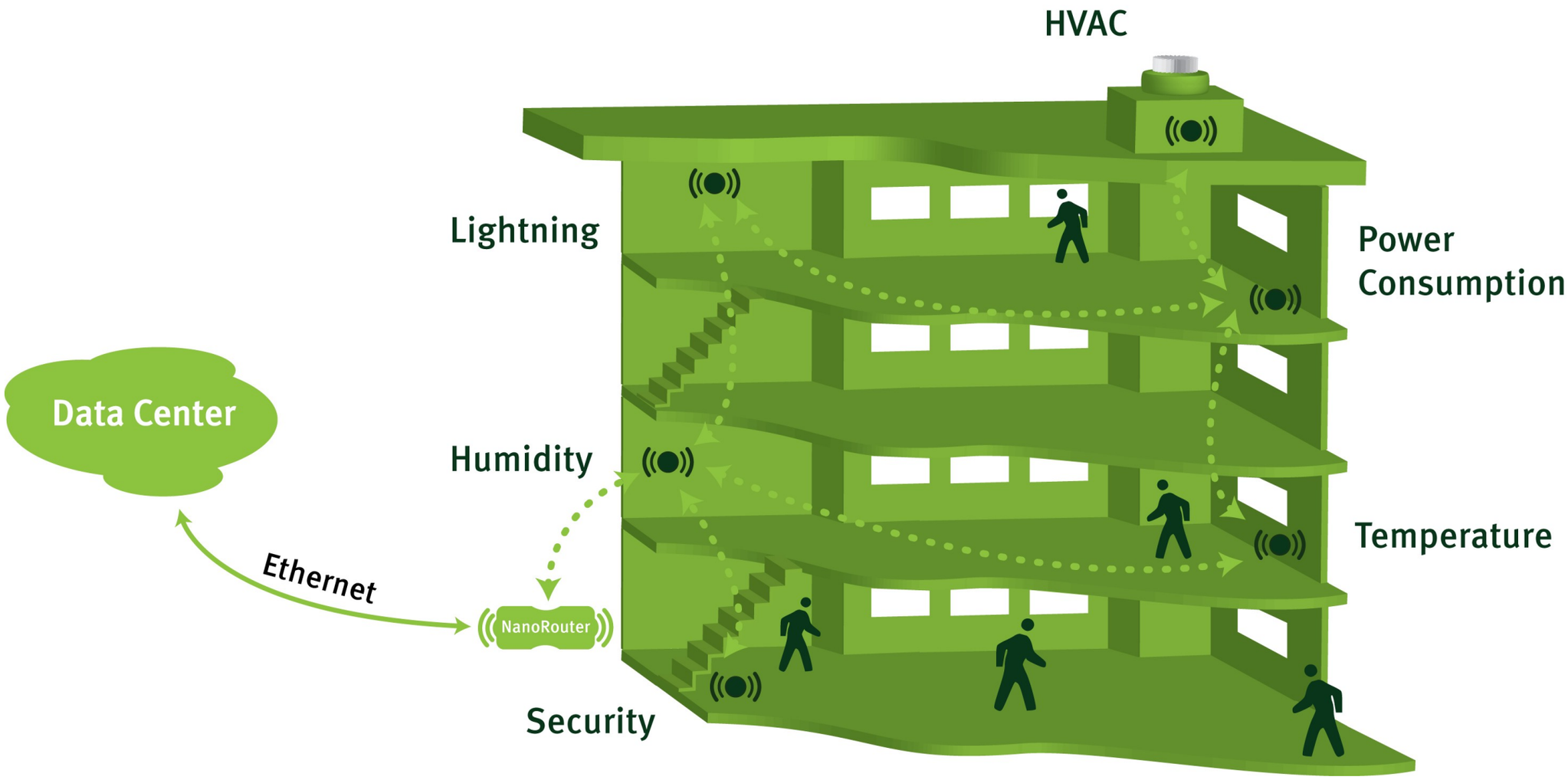
Example Applications

Smart Energy & Lighting

sensinode

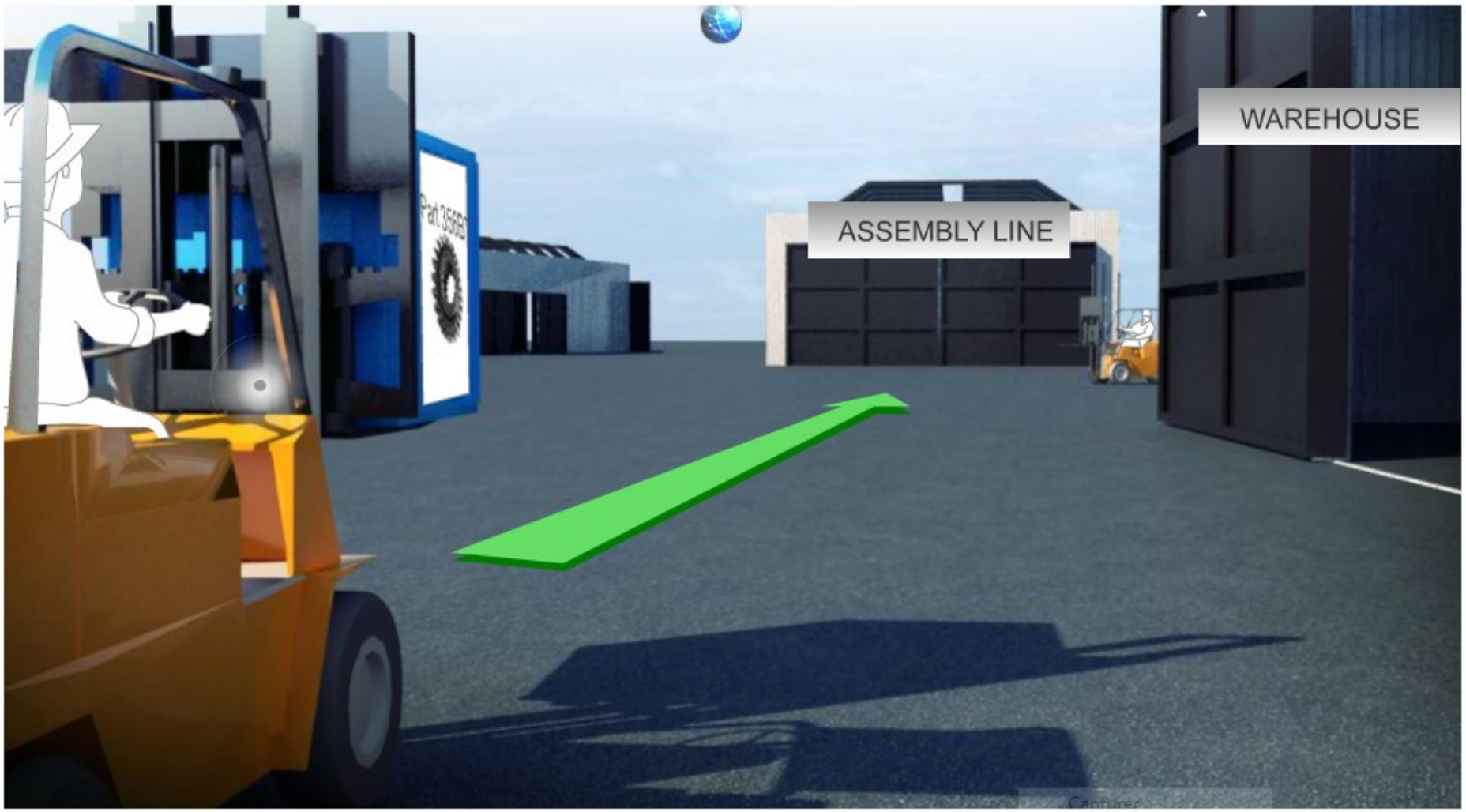


Building Automation



Health & Fitness

Asset Management

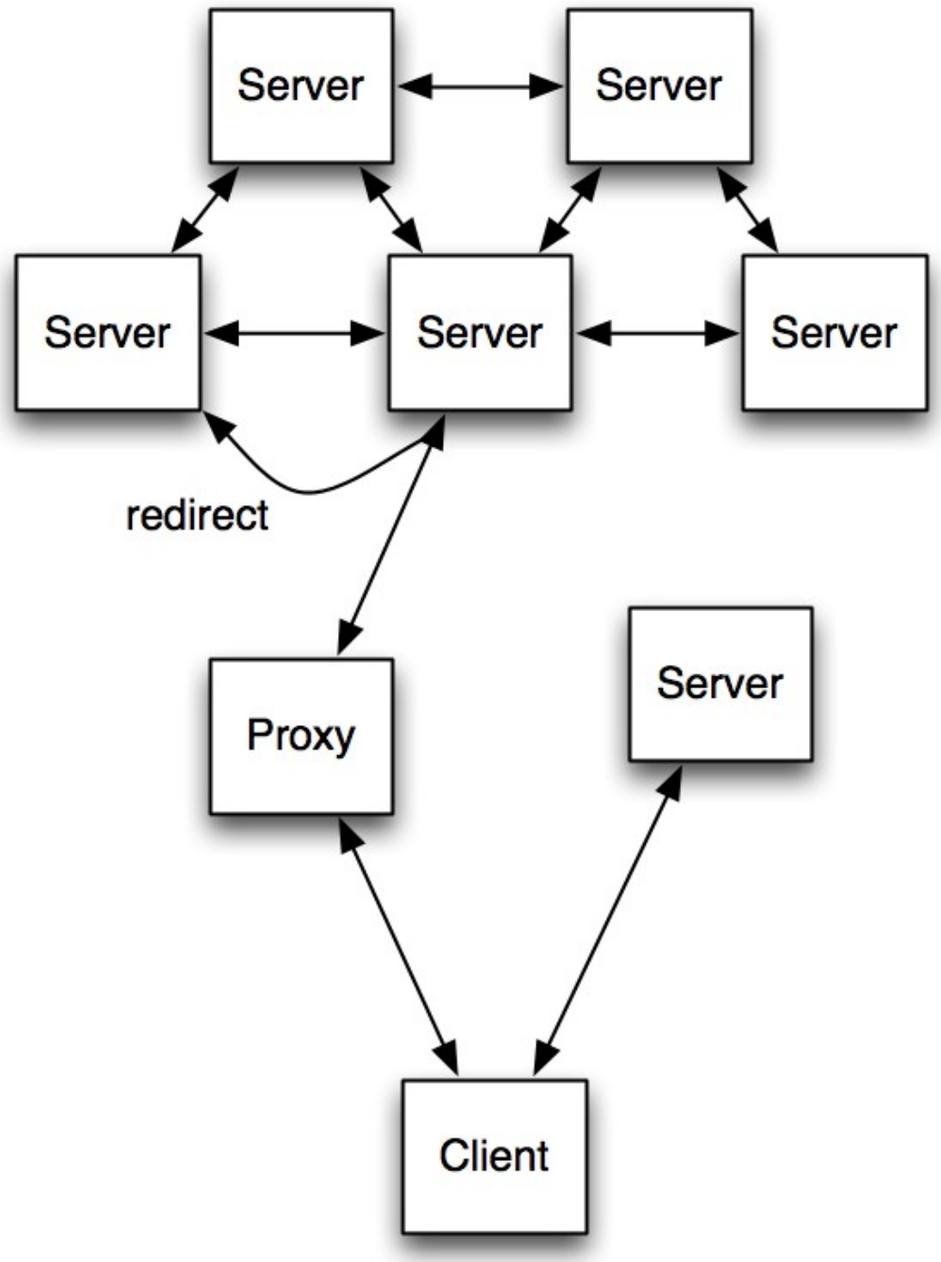


sensinode

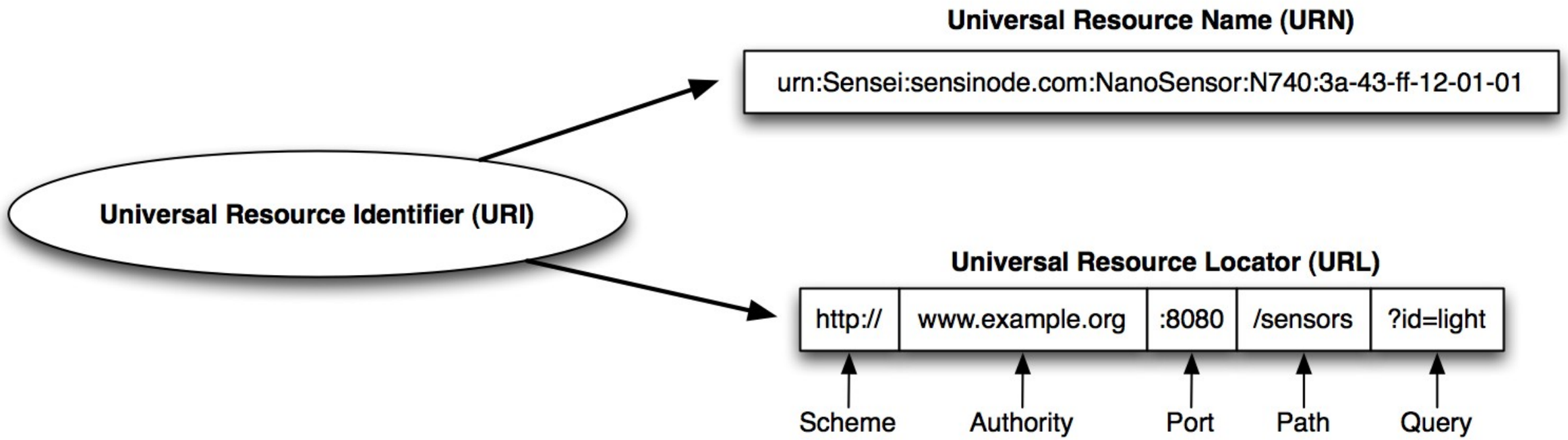
The Web and REST

The Web Architecture

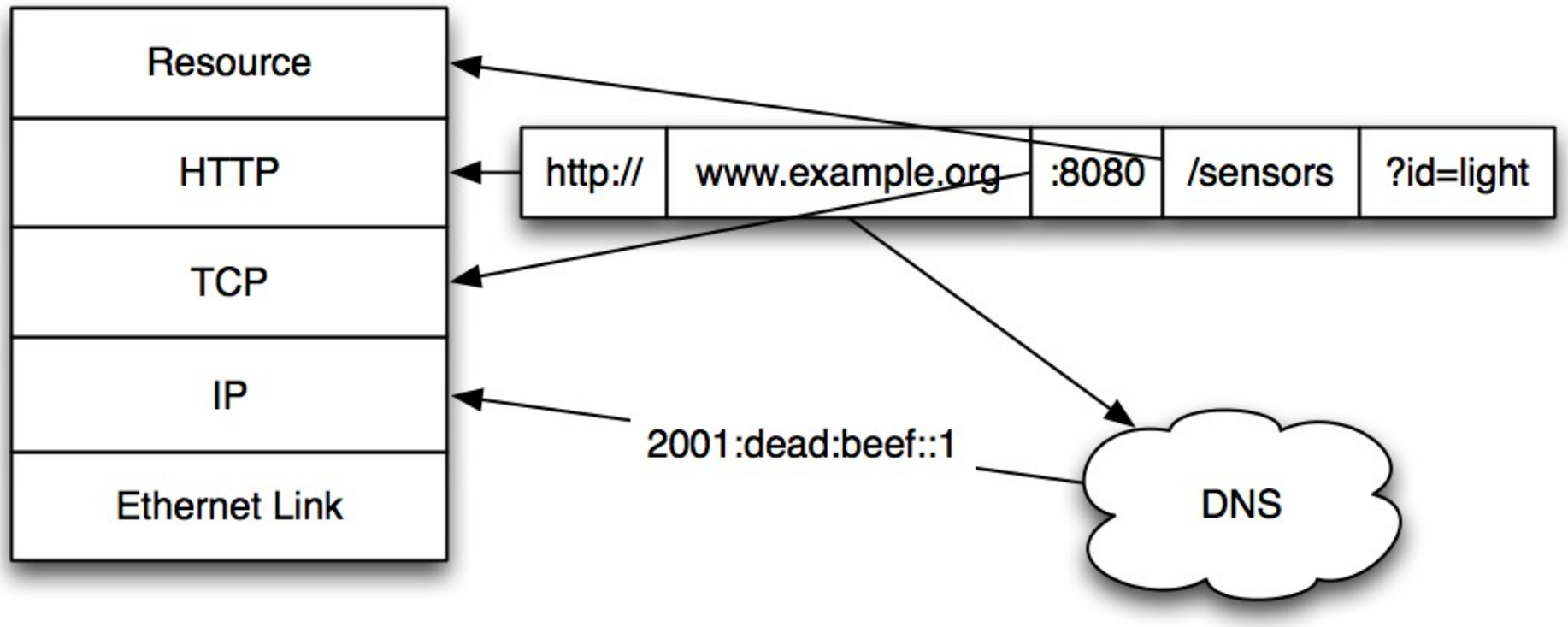
sensinode



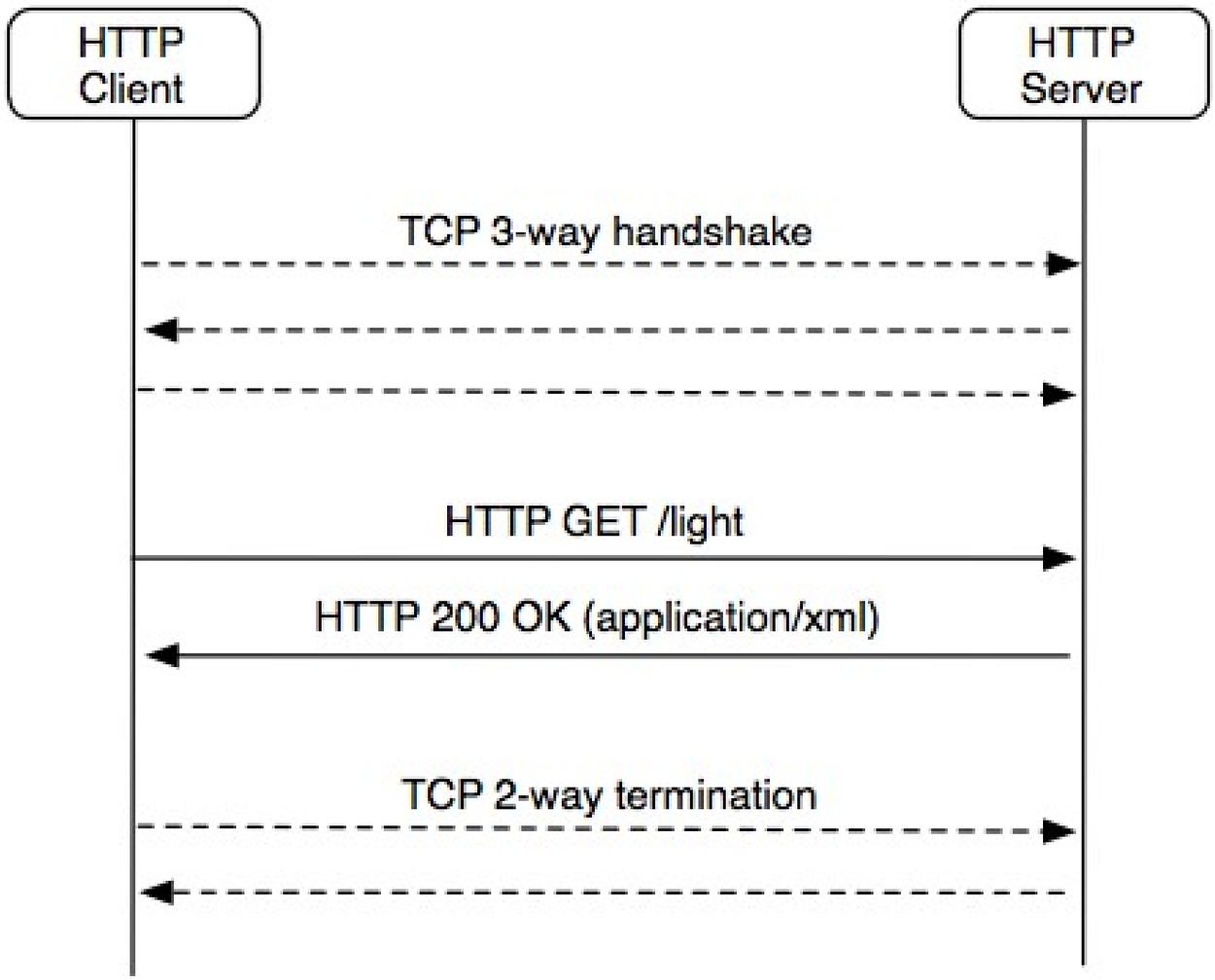
Web Naming


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

URL Resolution

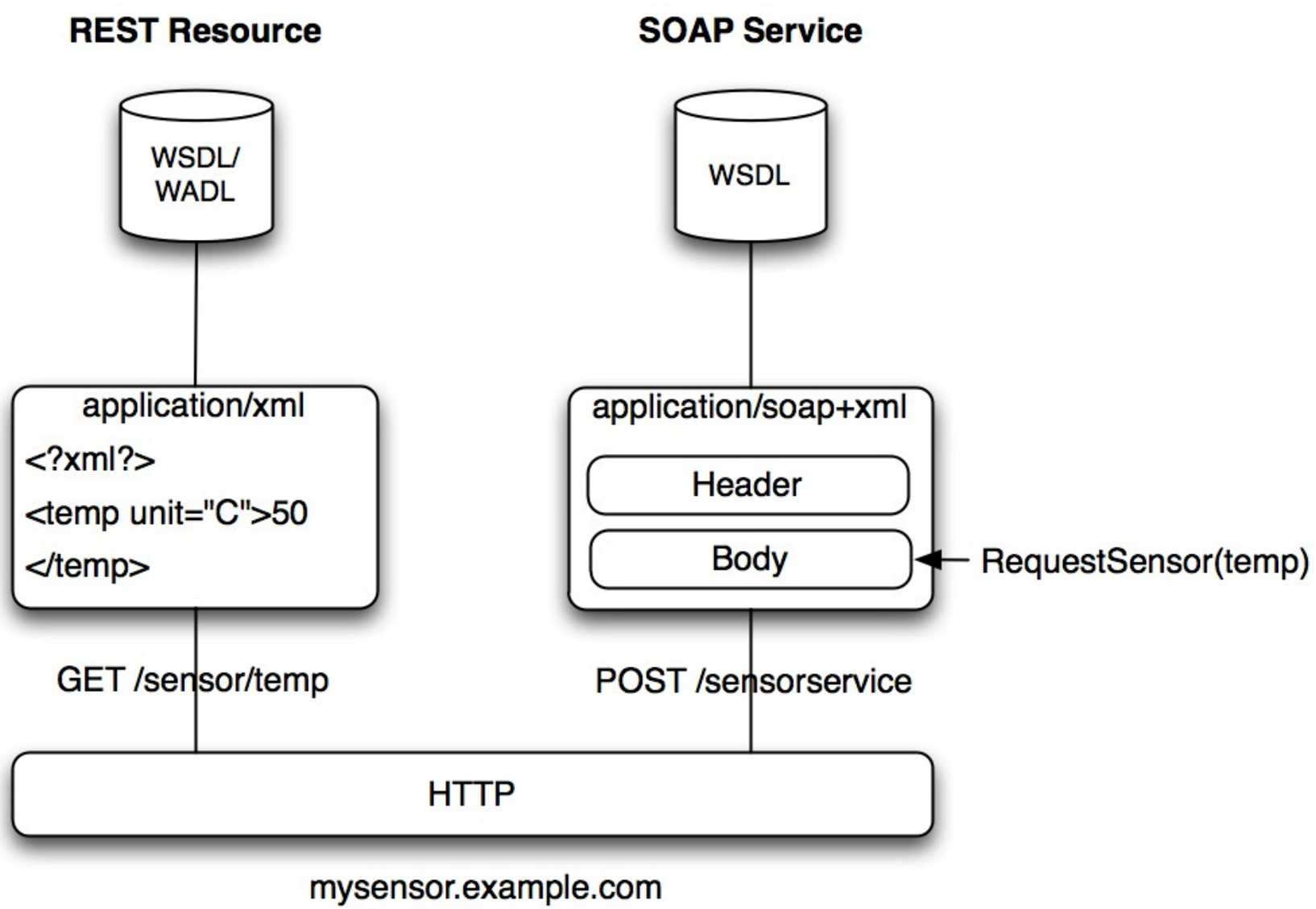


An HTTP Request



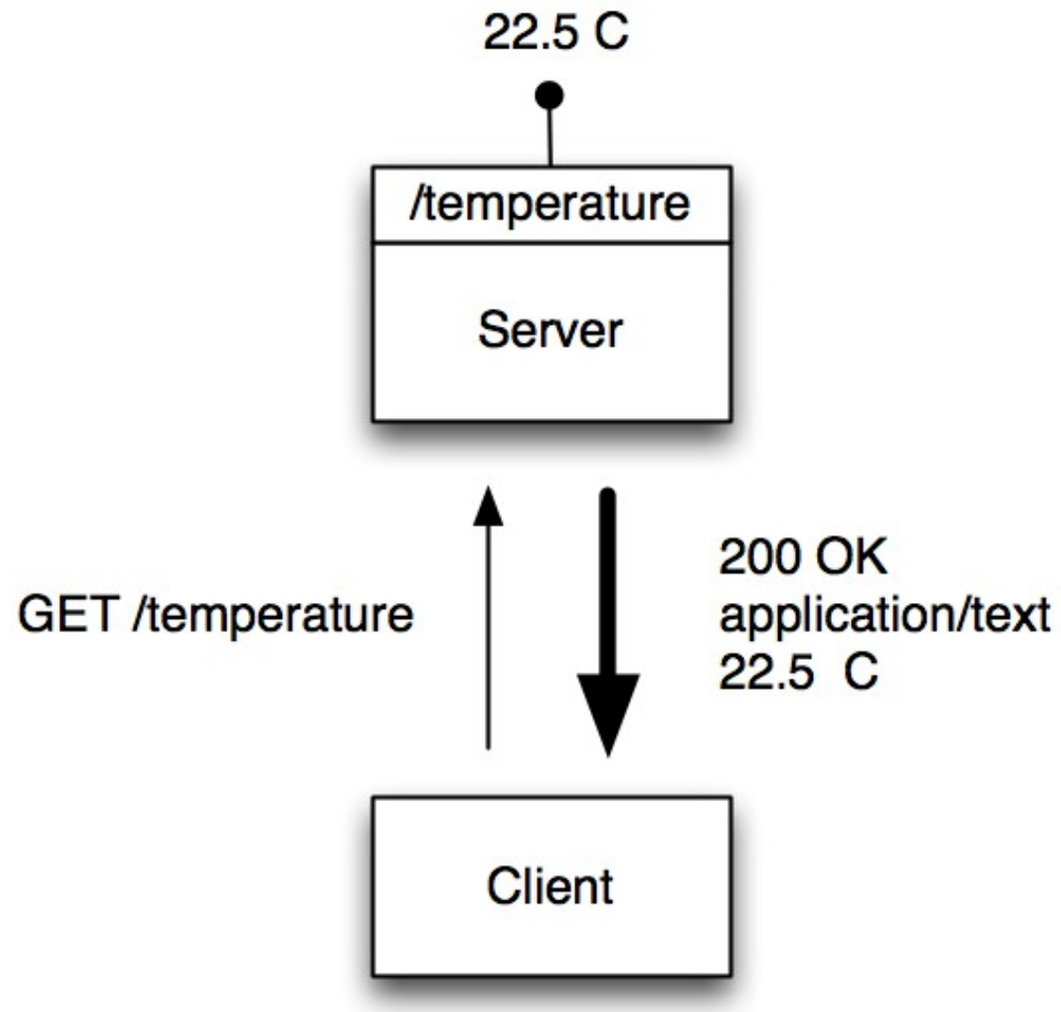
See RFC2616 - Hypertext Transfer Protocol v1.1

Web Paradigms



A REST Request

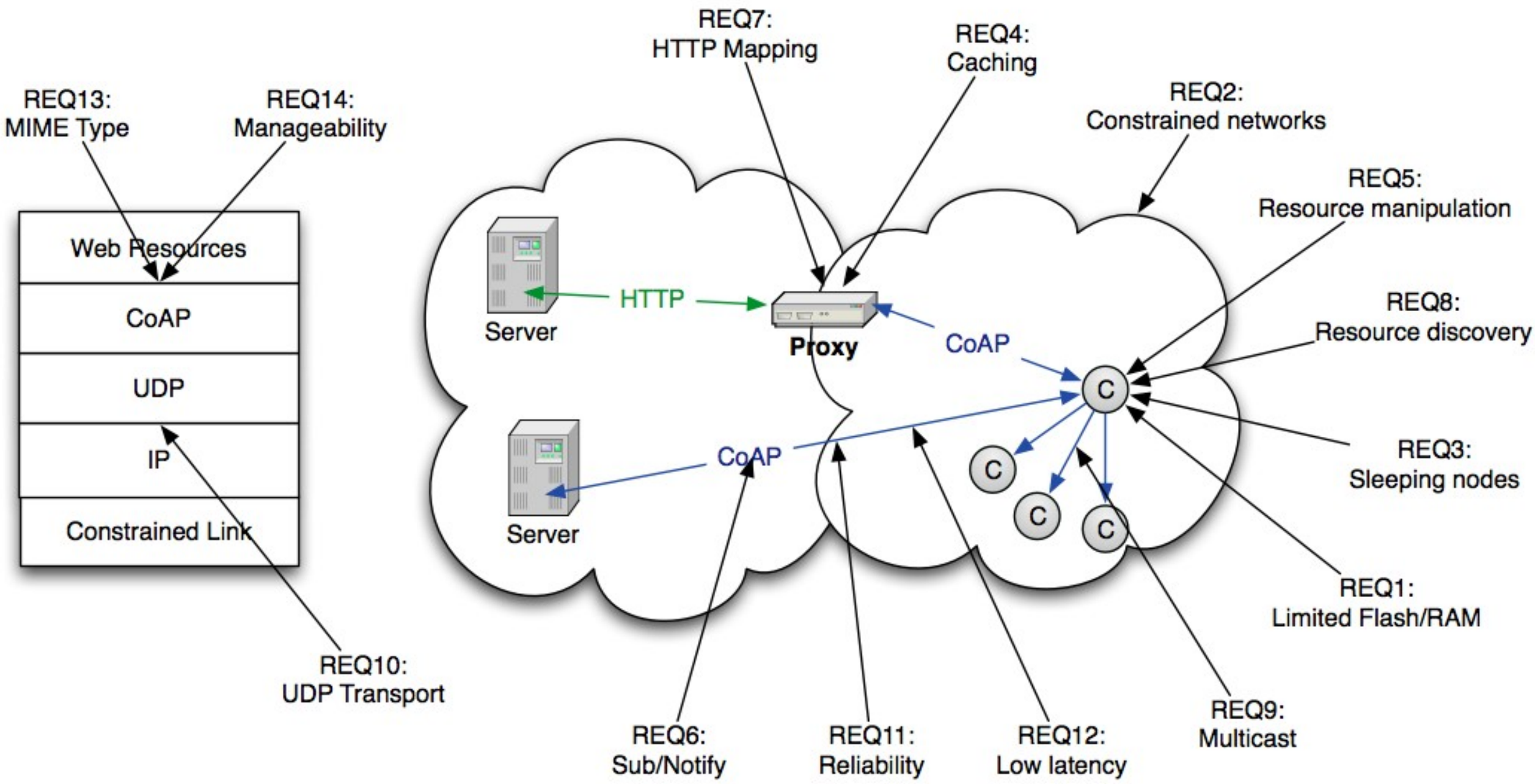
sensinode



sensinode

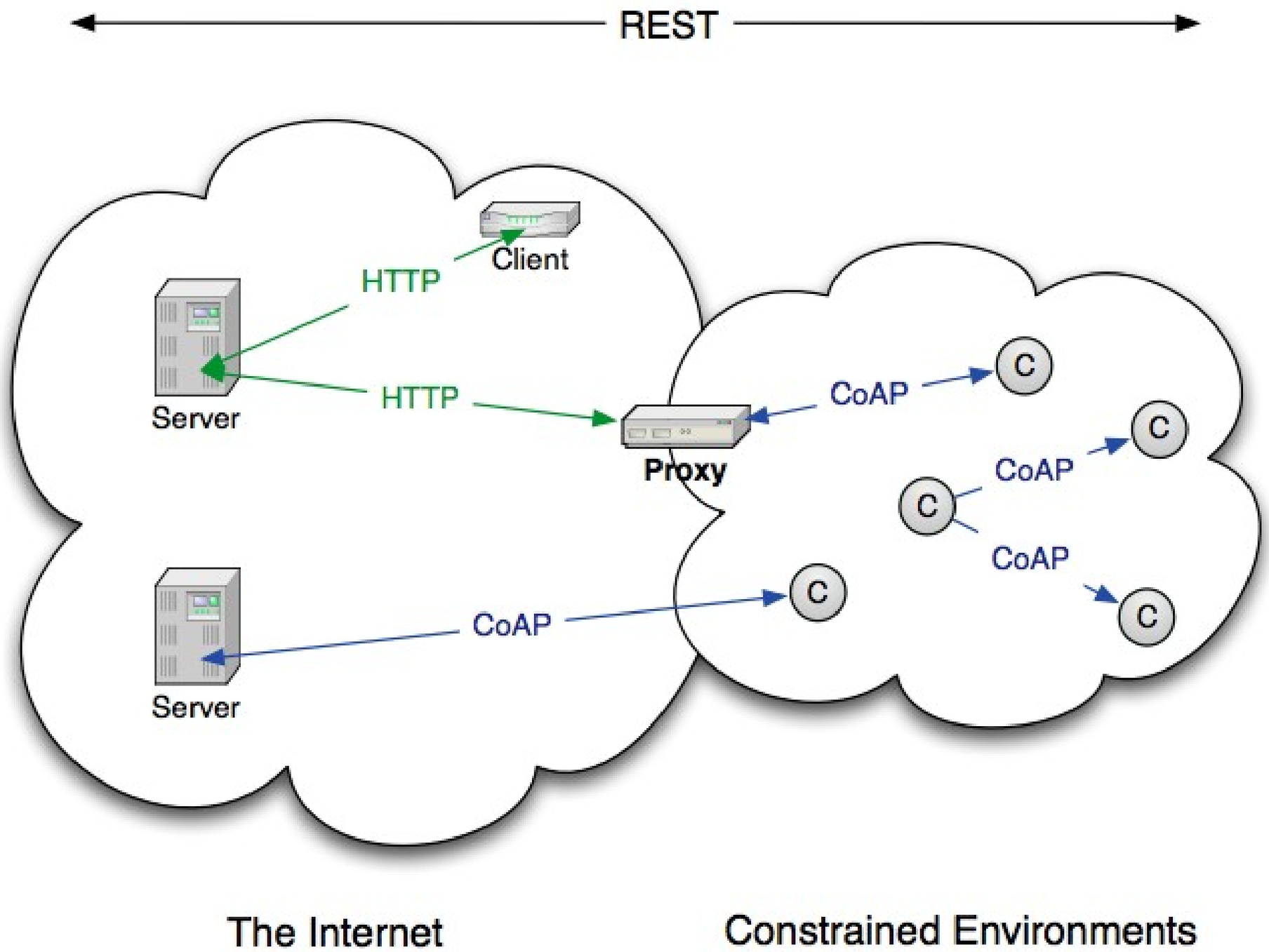
CoAP - Constrained Application Protocol

CoAP Design Requirements



See draft-shelby-core-coap-req

The CoAP Architecture



What CoAP is (and is not)

sensinode

- 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

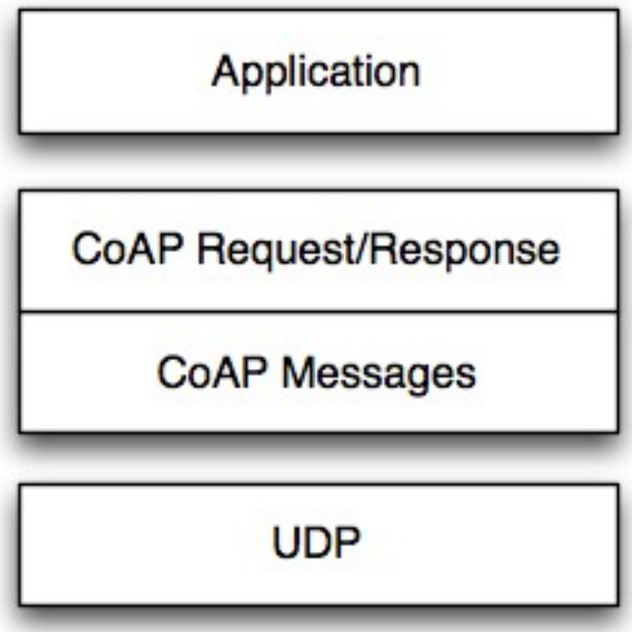
CoAP Features

sensinode

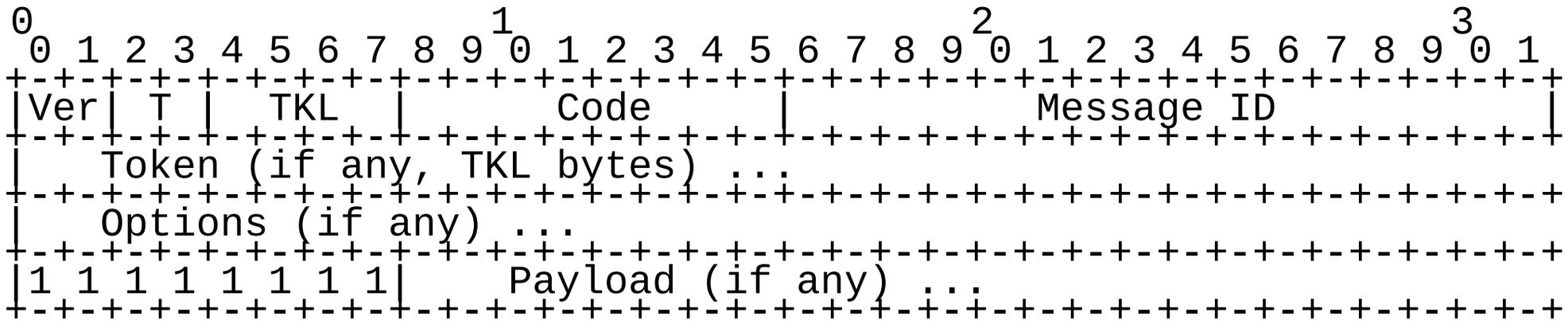
- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple 4 byte header
- DTLS based PSK, RPK and Certificate security
- Subset of MIME types and HTTP response codes
-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

The 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)

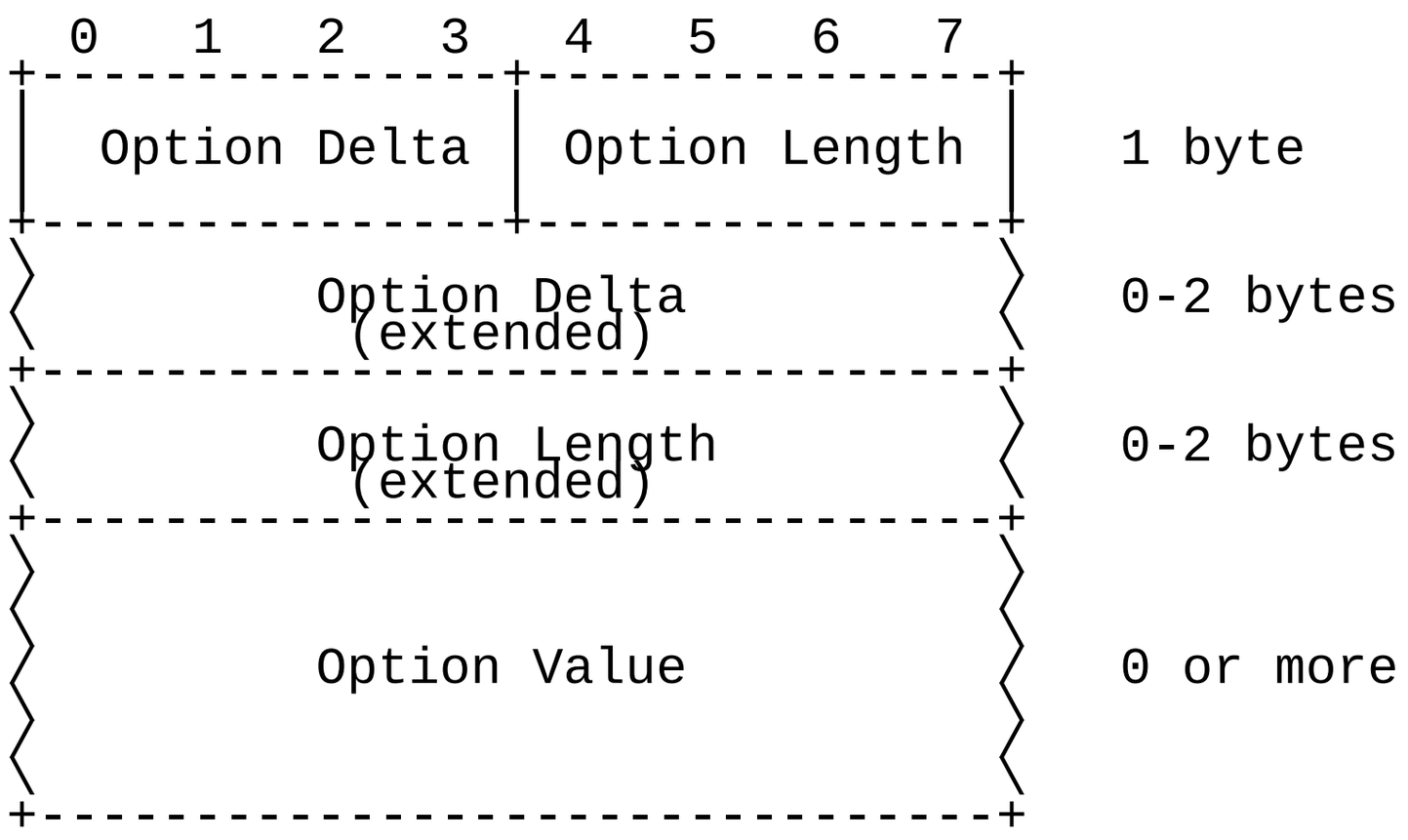
TKL- Token Length, if any, the number of Token bytes after this header

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



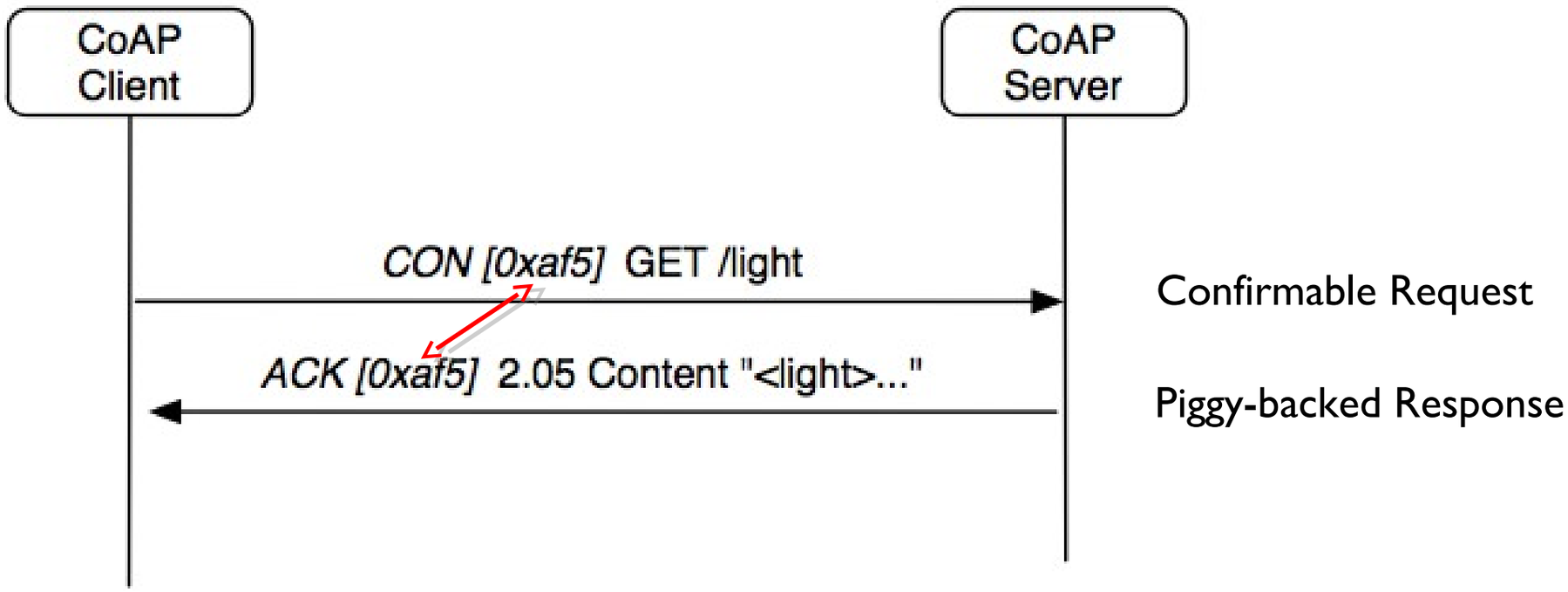
Option Delta - Difference between this option type and the previous
Length - Length of the option value
Value - The value of Length bytes immediately follows Length

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

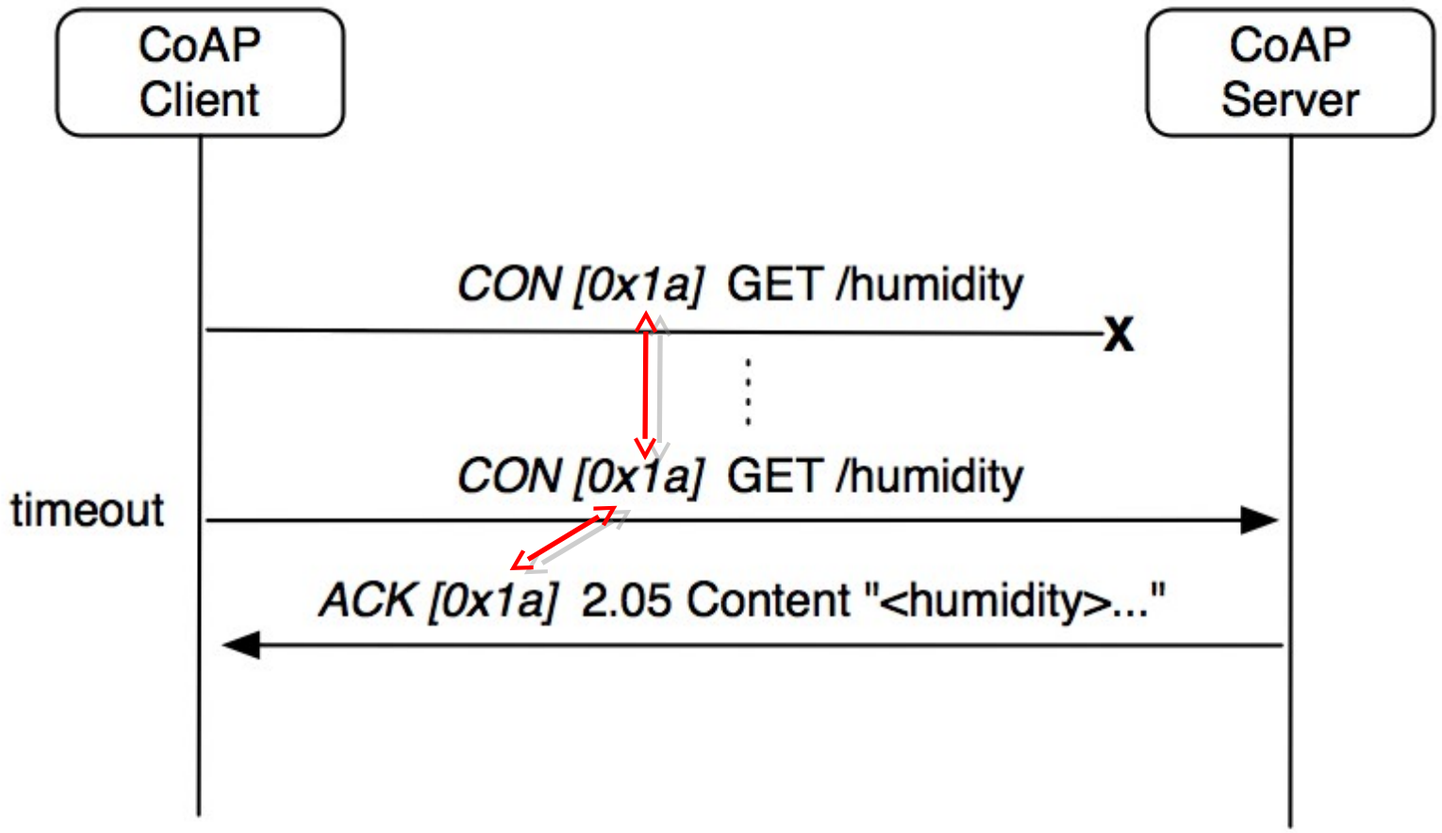
Request Example



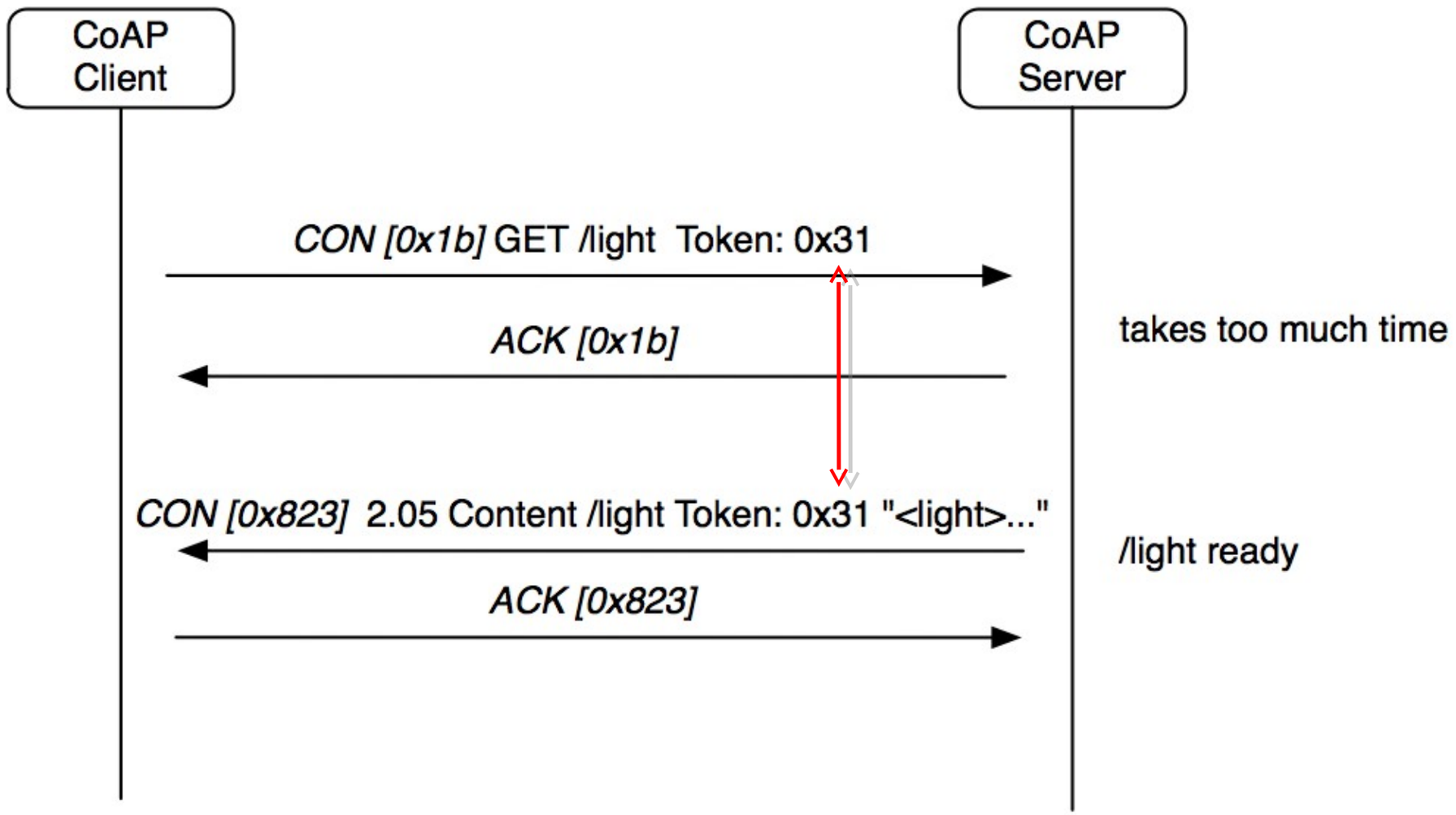
Confirmable Request

Piggy-backed Response

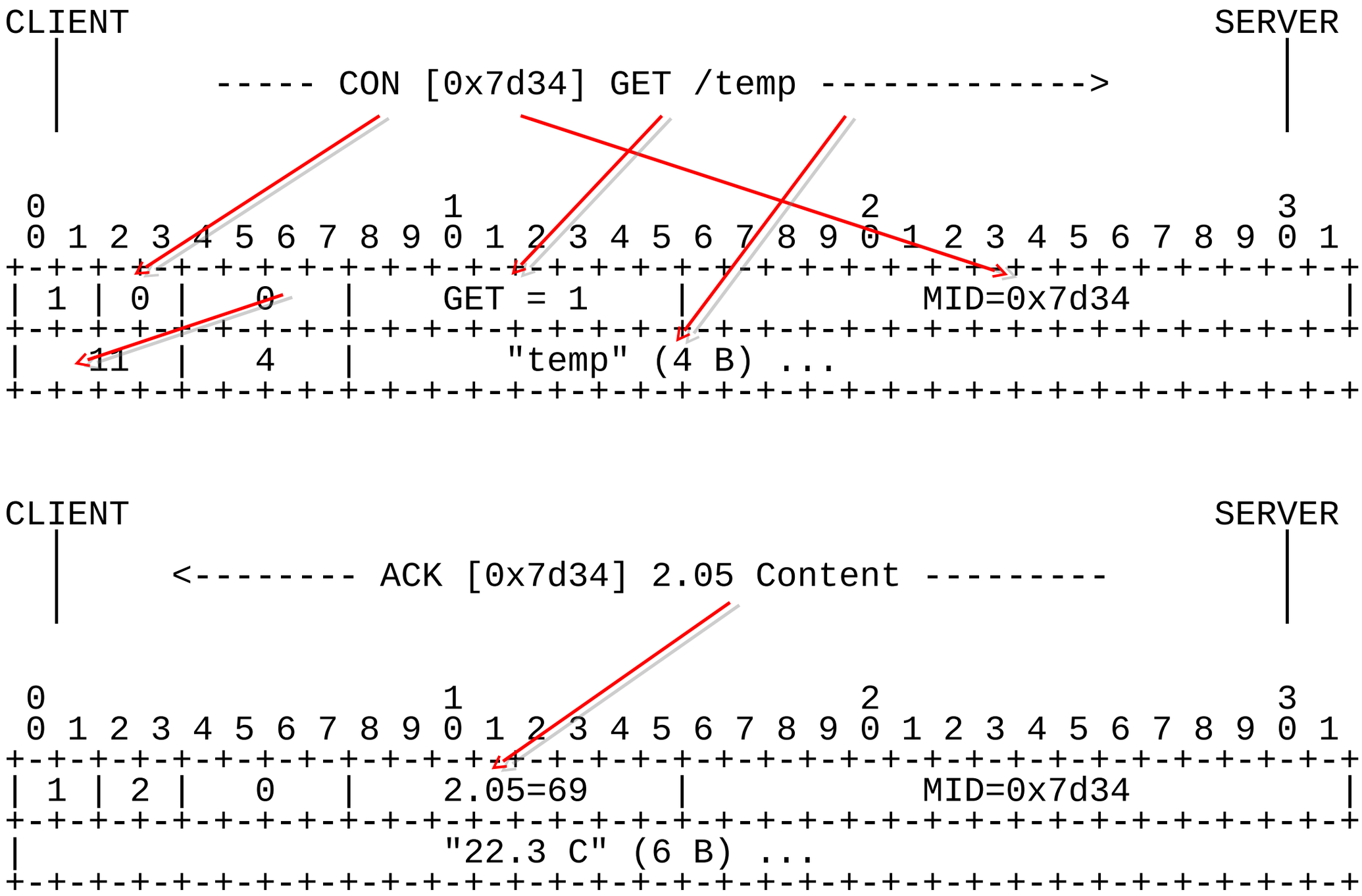
Dealing with Packet Loss



Separate Response



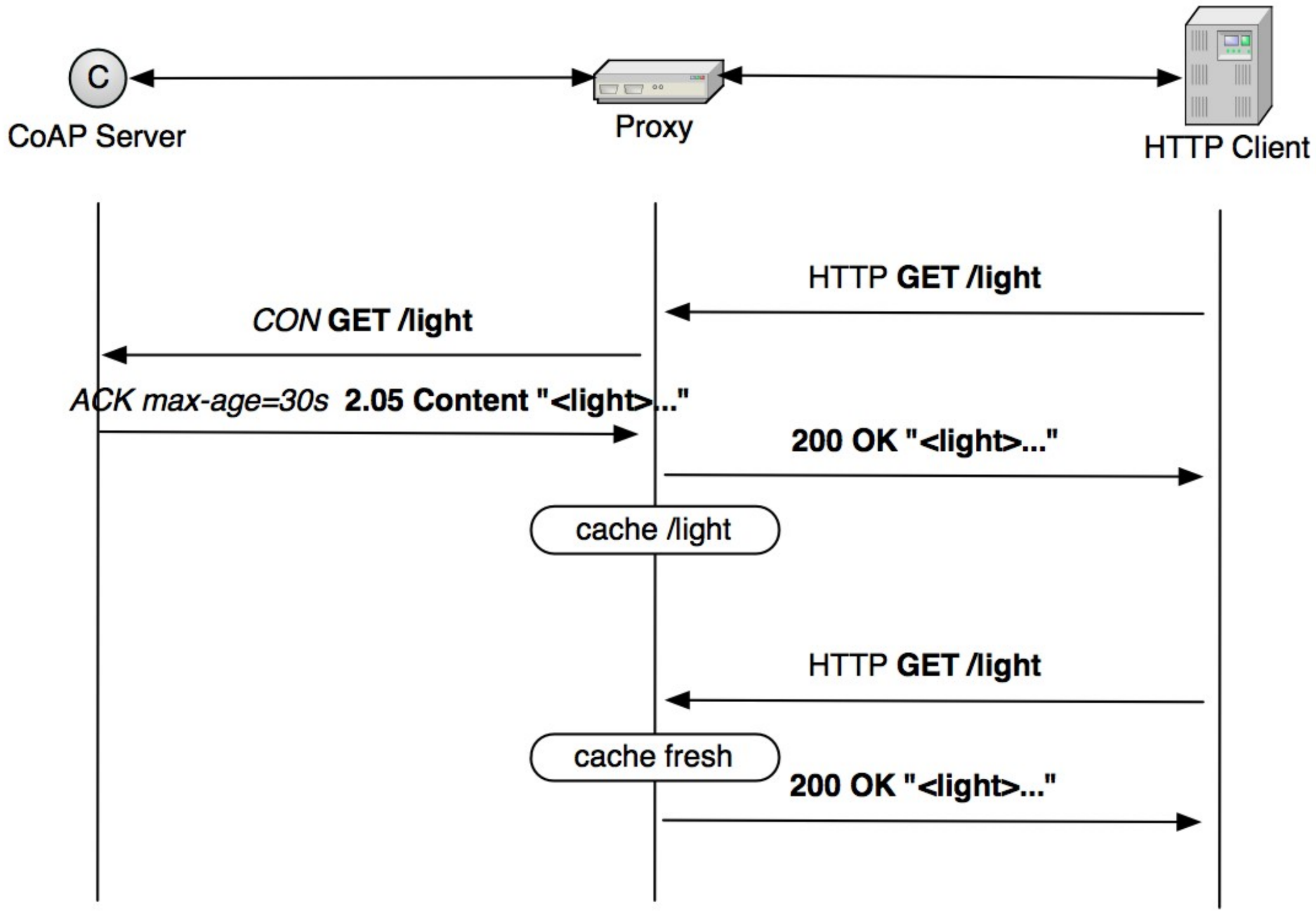
Bits and bytes...



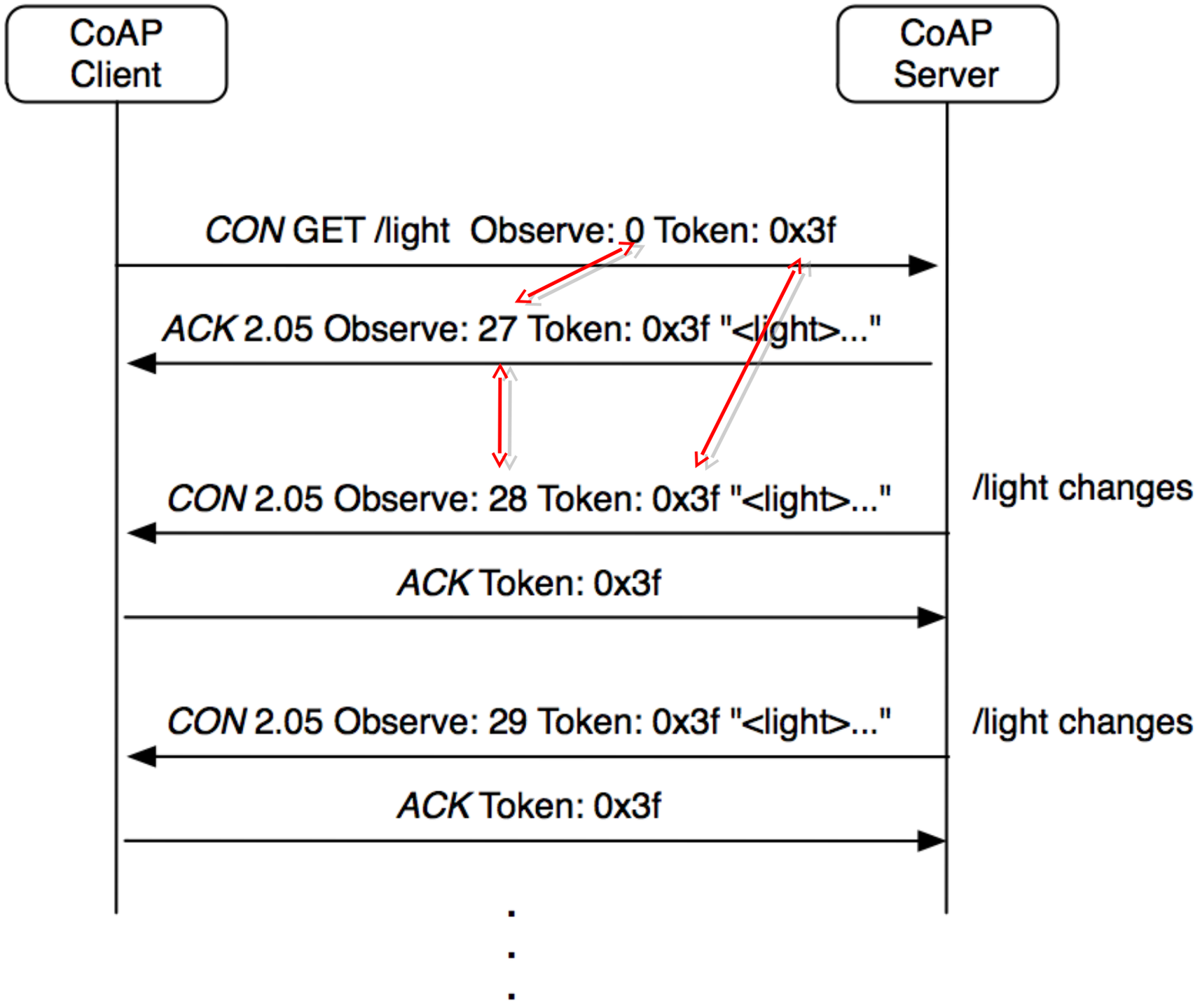
Caching

- 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

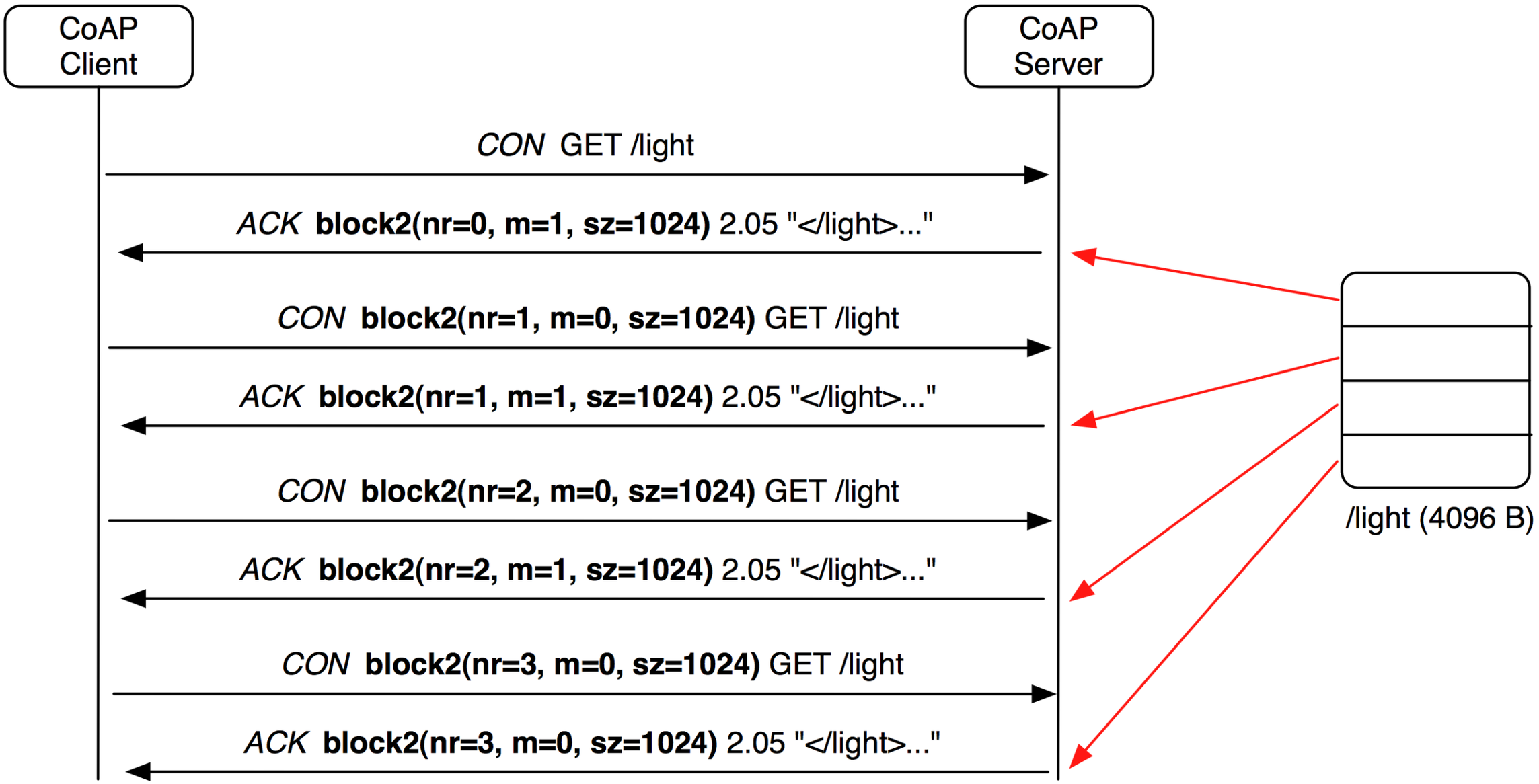


Observation



See draft-ietf-core-observe

Block transfer



See draft-ietf-core-block

Getting Started with CoAP

sensinode

- 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!

sensinode

Discovery & Semantics

What is Web Linking?

sensinode

- Links have been around a long time
- Web Linking formalizes links with defined relations, **typed links**
  - ✓ HTML and Atom have allow links
- RFC5988 defines a framework for Web Linking
  - ✓ Combines and expands the Atom and HTML relation types
  - ✓ Defines a unified typed link concept
- A link can be serialized in any number of formats
  - ✓ RFC5988 revives the HTTP Link Header and defines its format
  - ✓ Atom and HTML are equivalent serializations

What is Web Linking?

sensinode

- A type link consists of:
  - ✓ Context URI – What the link is from
  - ✓ Relation Type – Indicates the semantics of the link
  - ✓ Target URI – What the link is too
  - ✓ Attributes – Key value pairs describing the link or its target
- Relations include e.g. copyright, author, chapter, service etc.
- Attributes include e.g. language, media type, title etc.
- Example in HTTP Link Header format:

```
Link: <http://example.com/TheBook/chapter2>; rel="previous";
  title="previous chapter"
```

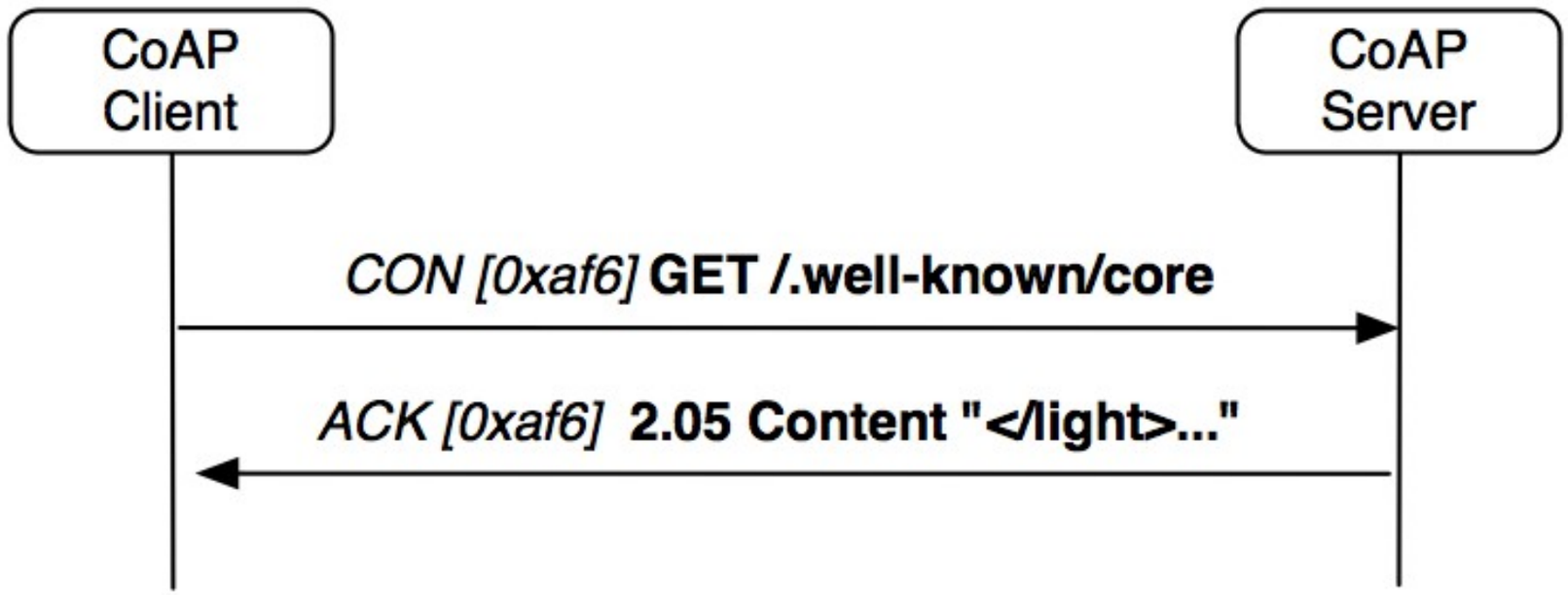
Resource Discovery

- Service Discovery
  - ✓ What services are available in the first place?
  - ✓ Goal of finding the IP address, port and protocol
  - ✓ Usually performed by e.g. DNS-SD when DNS is available
- Resource Discovery
  - ✓ What are the Web resources I am interested in?
  - ✓ Goal of finding URIs
  - ✓ Performed using Web Linking or some REST interface
    - ✓ CoRE Link Format is designed to enable resource discovery

CoRE Link Format

- RFC6690 is aimed at Resource Discovery for M2M
  - ✓ Defines a link serialization suitable for M2M
  - ✓ Defines a well-known resource where links are stored
  - ✓ Enables query string parameters for filtered GETs
  - ✓ Can be used with unicast or multicast (CoAP)
- Resource Discovery with RFC6690
  - ✓ Discovering the links hosted by CoAP (or HTTP) servers
  - ✓ GET /.well-known/core?optional_query_string
  - ✓ Returns a link-header style format
    - ✓ URL, relation, type, interface, content-type etc.

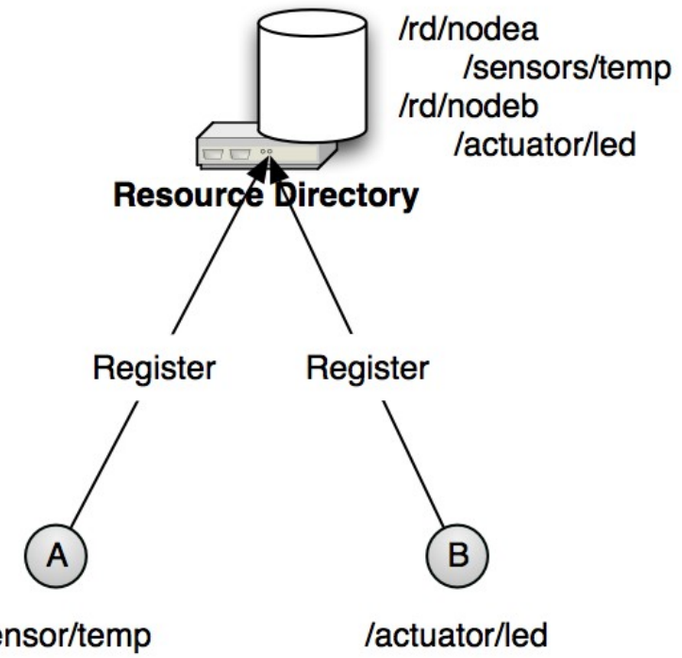
CoRE Resource Discovery




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

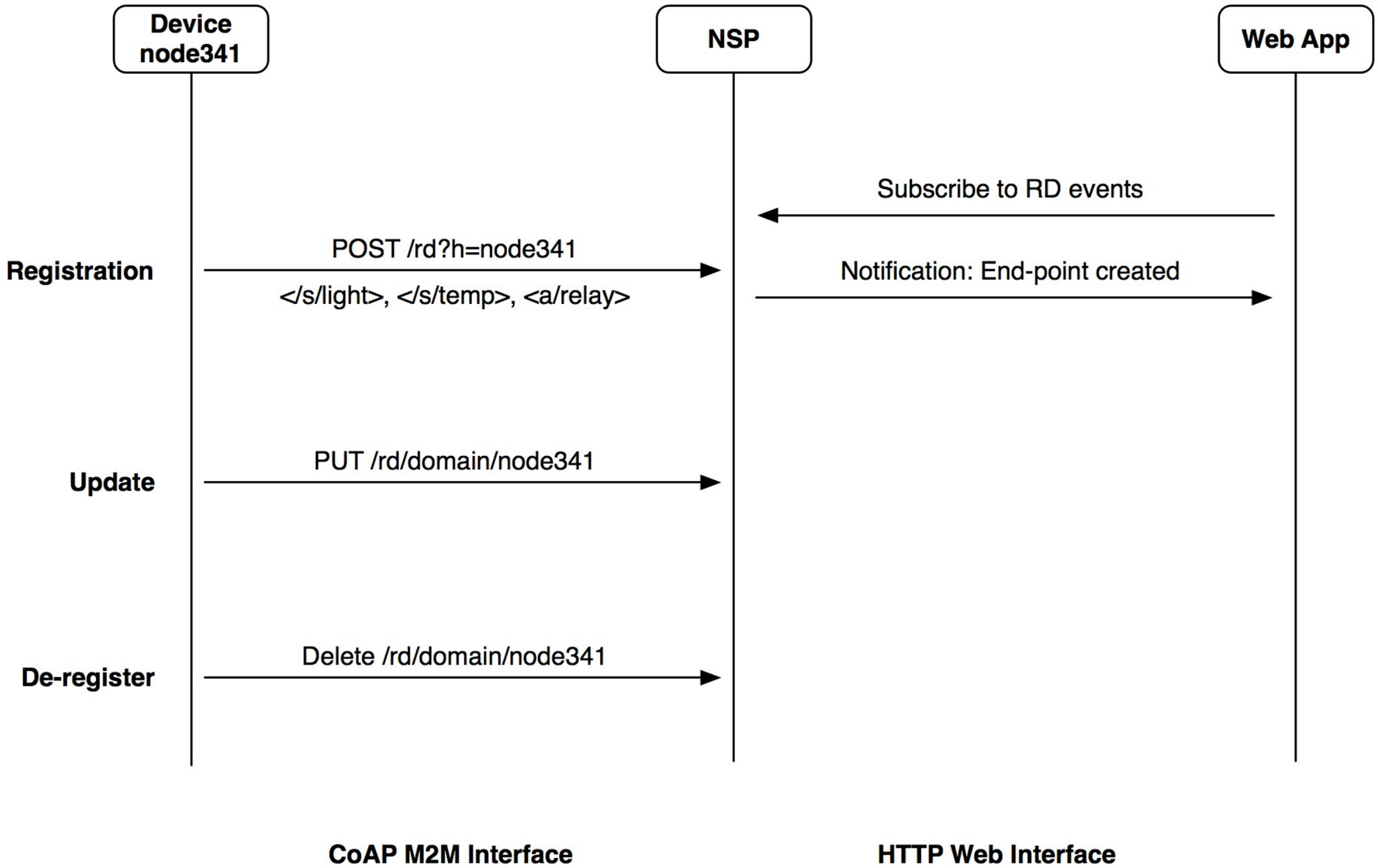
Resource Directory

- CoRE 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



See draft-shelby-core-resource-directory

Resource Directory



See draft-shelby-core-resource-directory

How to get Semantic?

- So how to use CoRE in real applications?
- Resources need meaningful naming (rt=)
- A resource needs an interface (if=)
  - ✓ See [draft-vial-core-link-format-wadl] on using WADL for this
- A payload needs a format (EXI, JSON etc.)
  - ✓ Deployment or industry specific today
  - ✓ oBIX, SensorML, EEML, sMAP etc.
  - ✓ SenML is a promising format [draft-jennings-senml]
- What can we make universal?
- What should be market specific?
- How do we enable innovation?

CoRE Link Format Semantics

- RFC6690 = Simple semantics for machines
  - ✓ IANA registry for rt= and if= parameters
- Resource Type (rt=)
  - ✓ What is this resource and what is it for?
  - ✓ e.g. Device Model could be rt="ipso.dev.mdl"
- Interface Description (if=)
  - ✓ How do I access this resource?
  - ✓ e.g. Sensor resource accessible with GET if="core.s"
- Content Type (ct=)
  - ✓ What is the data format of the resource payloads?
  - ✓ e.g. text/plain (0)

CoRE Interfaces

- CoRE Interfaces [draft-shelby-core-interfaces]
  - ✓ A paradigm for REST profiles made up of function sets
  - ✓ Simple interface types

Interface	if=	Methods
Link List	core.ll	GET
Batch	core.b	GET, PUT, POST (where applicable)
Linked Batch	core.lb	GET, PUT, POST, DELETE (where applicable)
Sensor	core.s	GET
Parameter	core.p	GET, PUT
Read-only Parameter	core.rp	GET
Actuator	core.a	GET, PUT, POST
Binding	core.bnd	GET, POST, DELETE

Benefits of OMA Lightweight M2M

- Simple, efficient protocol, interfaces and payload formats
- Banking class security based on DTLS
  - ✓ With Pre-shared and Public Key modes, Provisioning and Bootstrapping
- Powerful Object and Resource model
  - ✓ Global registry and public lookup of all Objects
  - ✓ Provides application semantics that are easy to use and re-use
  - ✓ Standard device management Objects already defined by OMA
- Applicable to Cellular, 6LoWPAN, WiFi and ZigBee IP or any other IP based constrained devices or networks
- Ideal time-to-market for the standard
  - ✓ LWM2M is commercially deployable in 2013
  - ✓ Can be combined with existing DM offerings
  - ✓ Will be supported in OneM2M and can be integrated with ETSI M2M

Architecture