sensinode

Nov 1st, 2011

Embedded Web Services

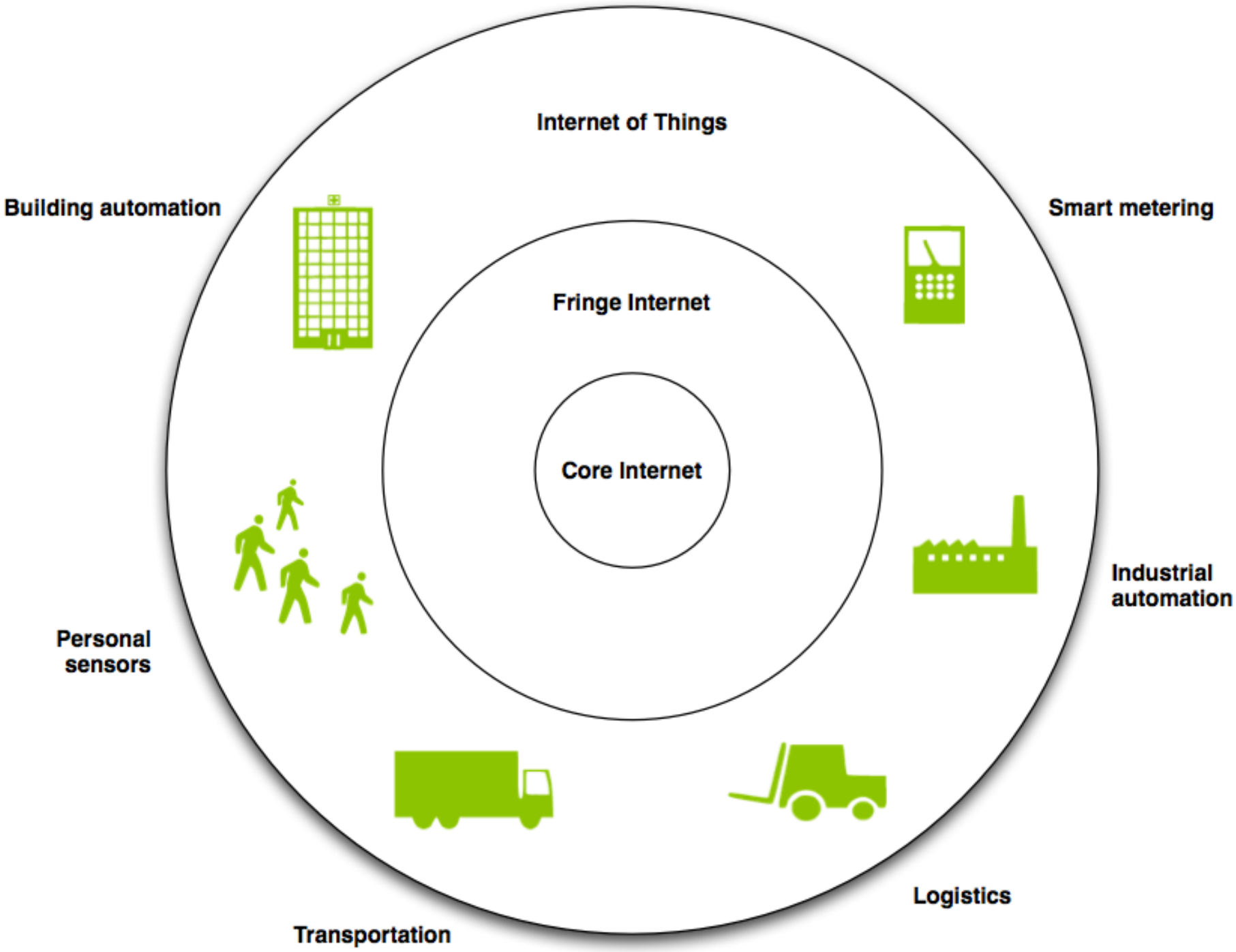
Zach Shelby, Chief Nerd

Course Overview

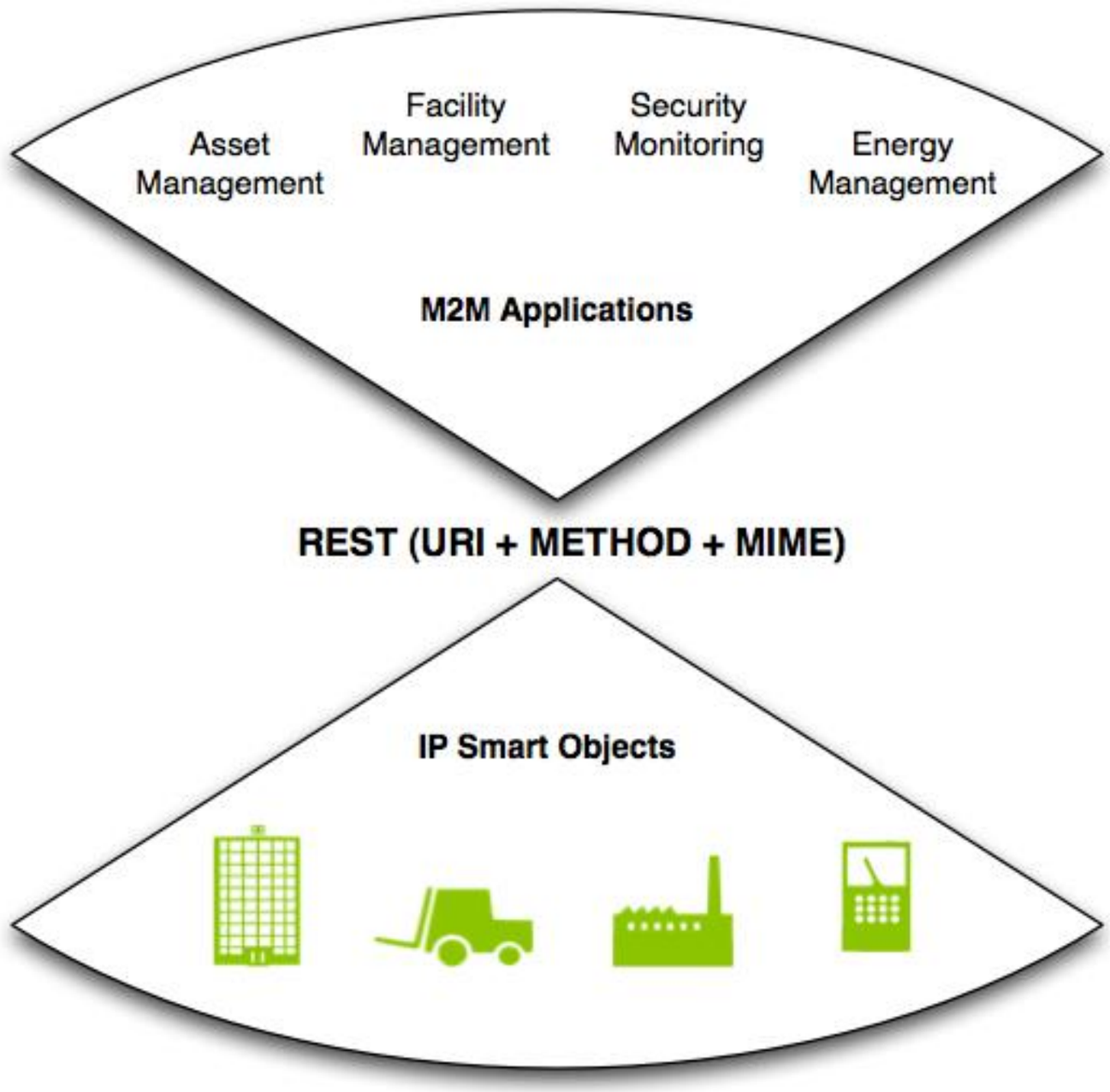
- Powering M2M with the Internet of Things
- Industry examples
- What are Web Services?
- CoRE - Constrained RESTful Environments
- Web Linking
- Resource-based M2M

The Internet of Things

sensinode



The Web of Things



IoT Standardization

- **IETF**
  - ✓ 6LoWPAN Working Group (L3)
  - ✓ ROLL (Routing Over Low-power Lossy Networks) WG
  - ✓ CoRE (Constrained RESTful Environments) WG
- **W3C**
  - ✓ Efficient XML Interchange (EXI) standardization
- **OASIS**
  - ✓ Open information exchange efforts such as oBIX and eMIX
- **ZigBee IP**
  - ✓ An open-standard 6LoWPAN stack for Smart Energy 2.0
- **ETSI**
  - ✓ Work on complete M2M system standardization (M2M TC)
- **ISO/IEC**
  - ✓ Wireless Sensor Network Group

Industry applications

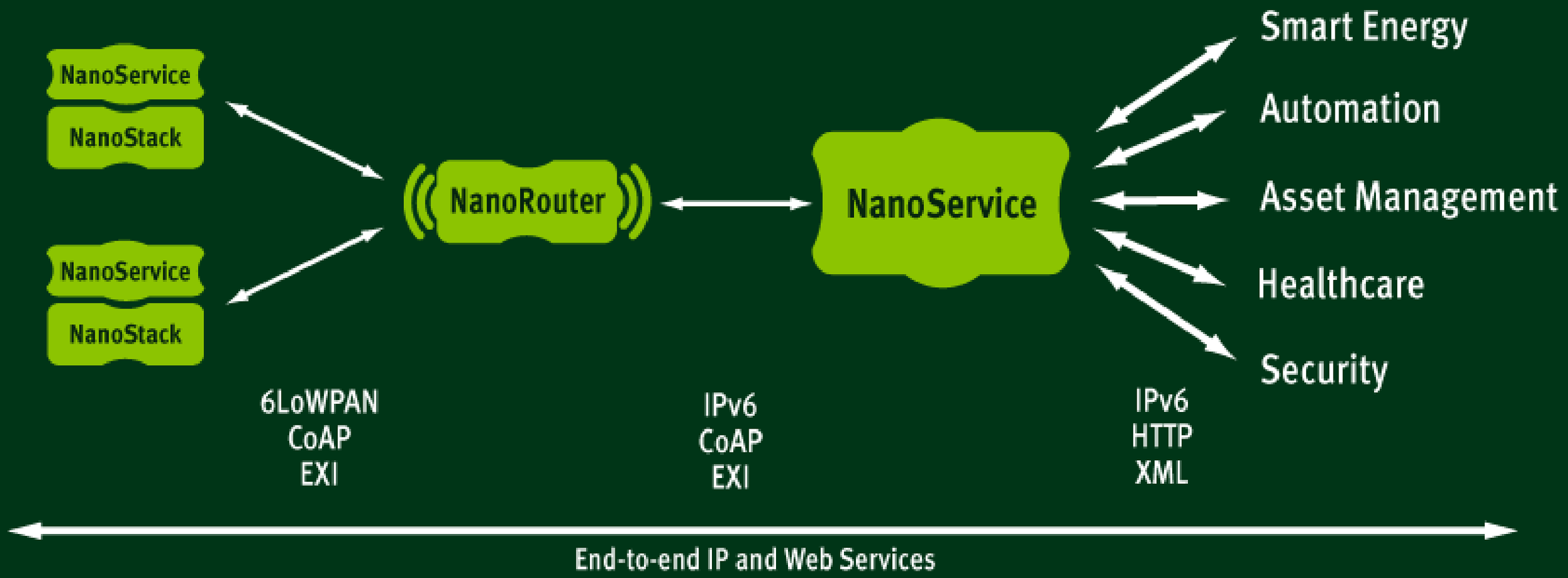
Sensinode NanoServices

sensinode

NanoStack™
Leading 6LoWPAN Stack

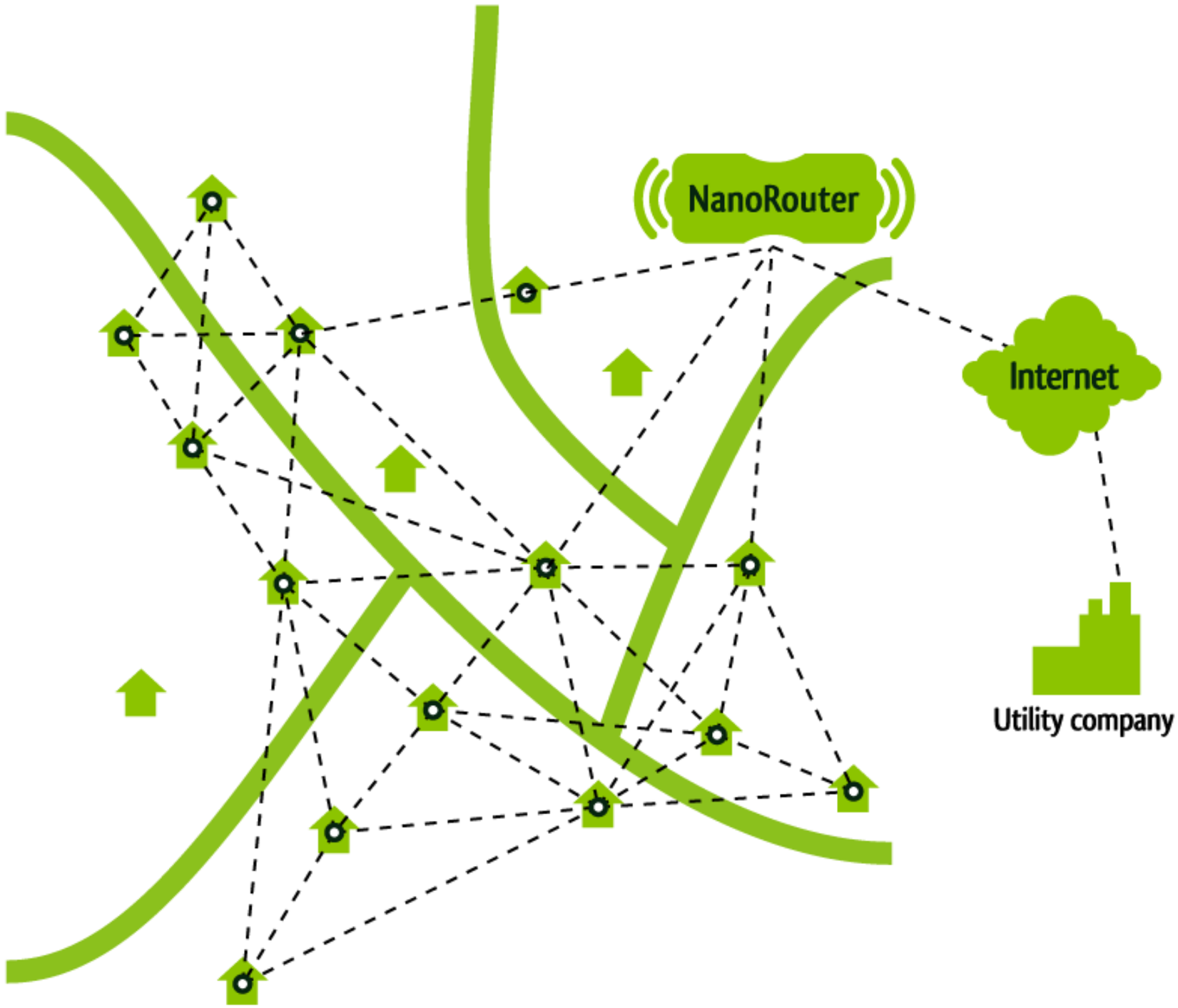
NanoRouter™
Seamless Internet connectivity

NanoService™
Web Services end-to-end

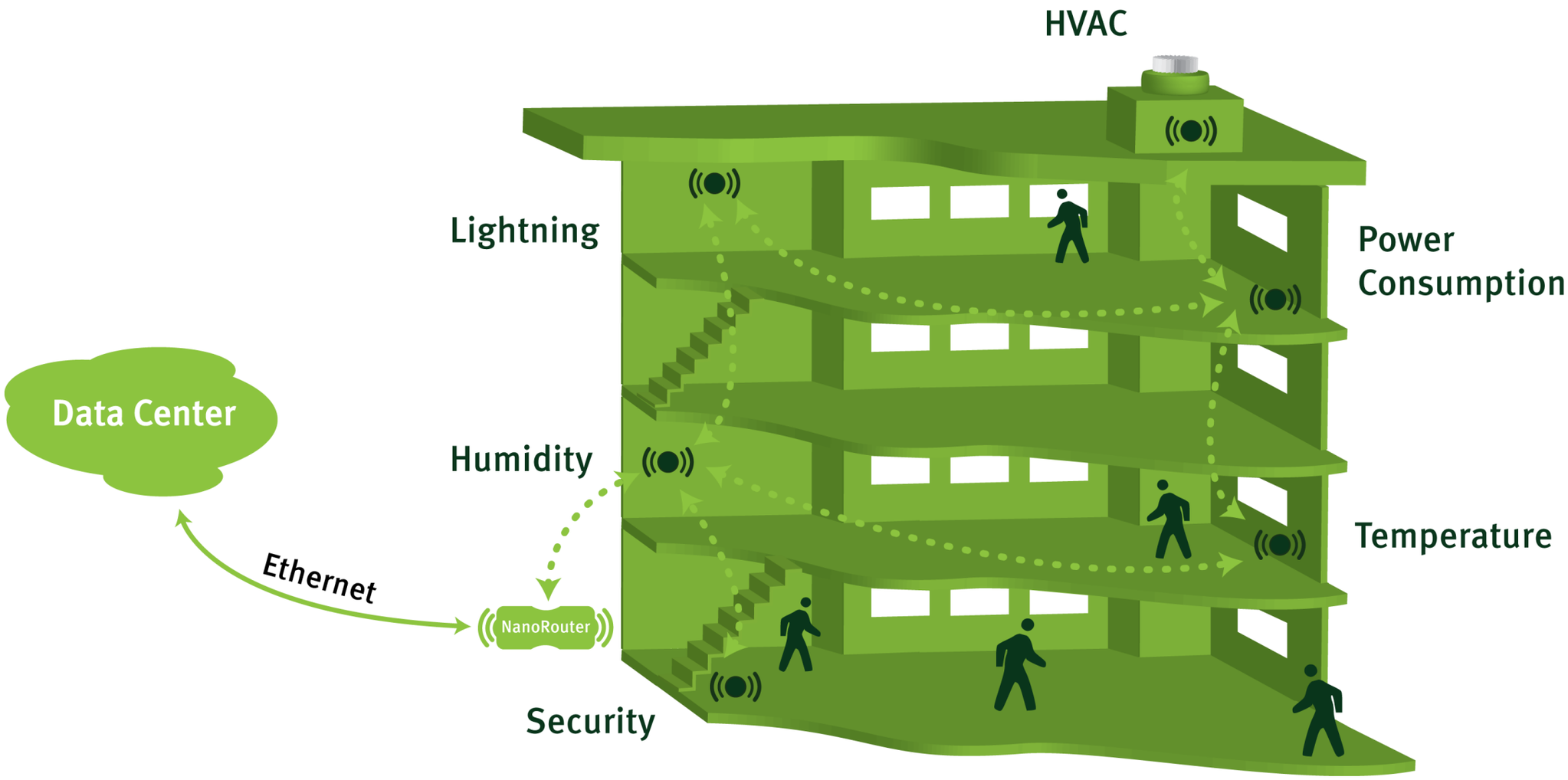


Smart Energy

sensinode

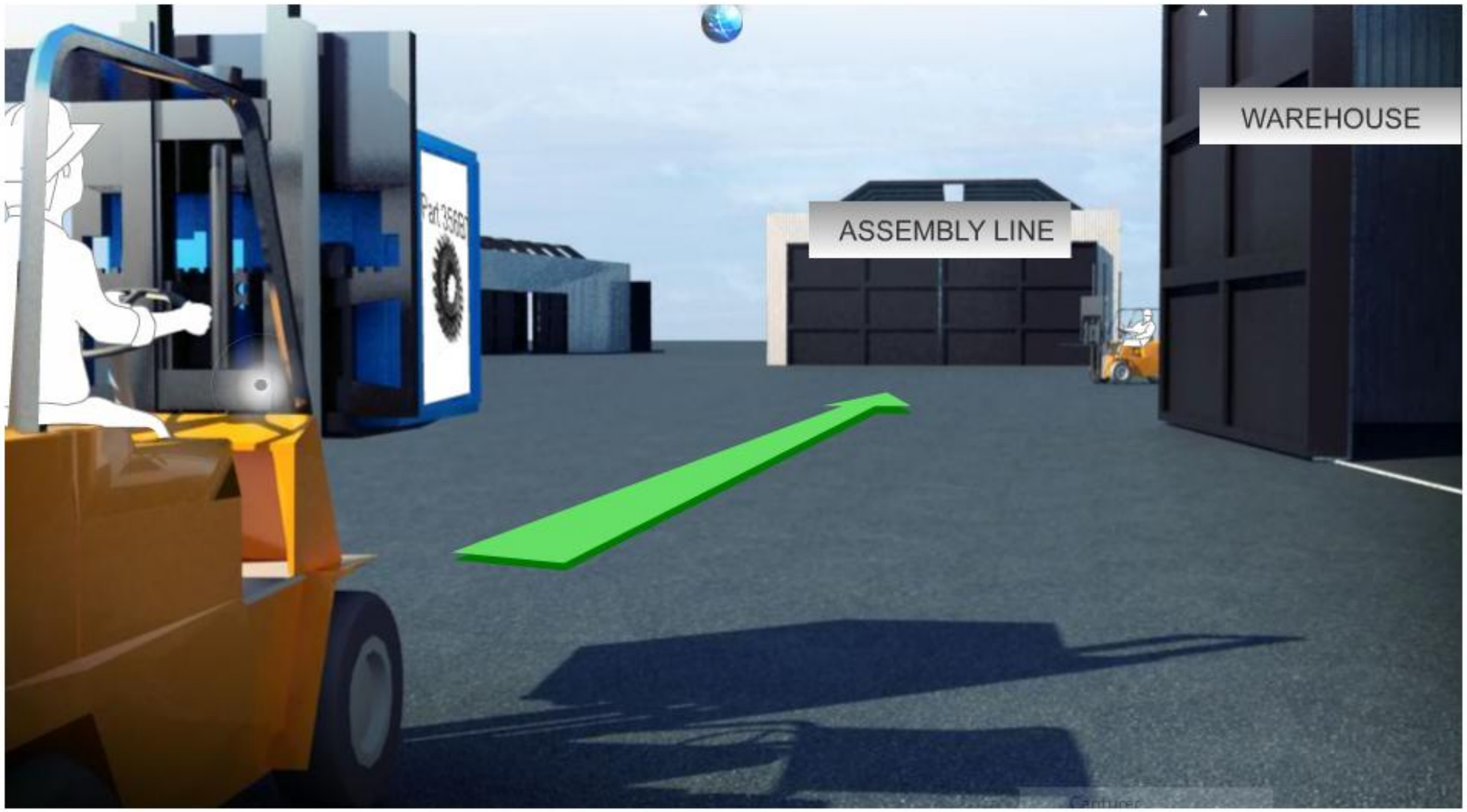


Building Automation



Fitness

Asset Management

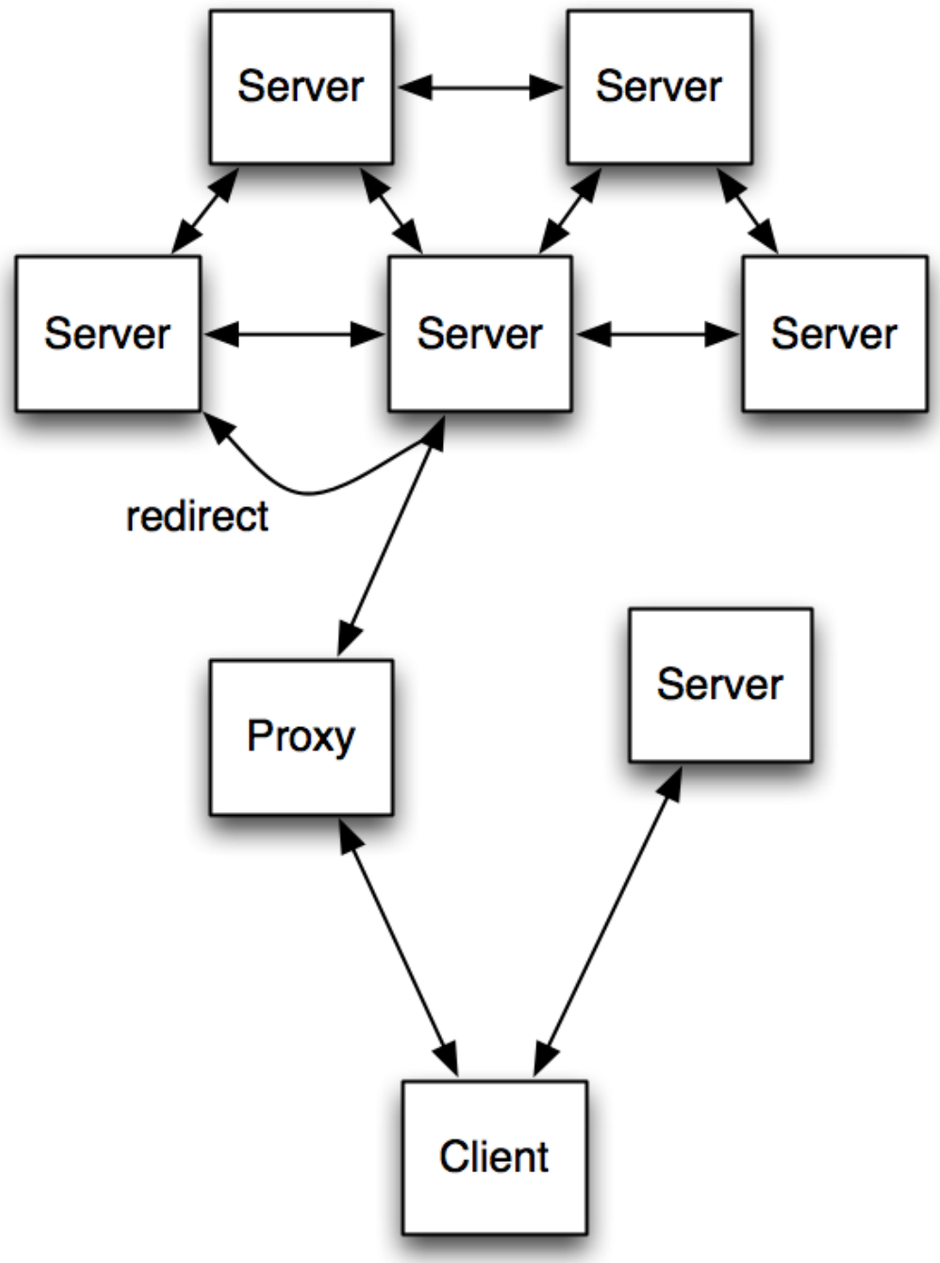


sensinode

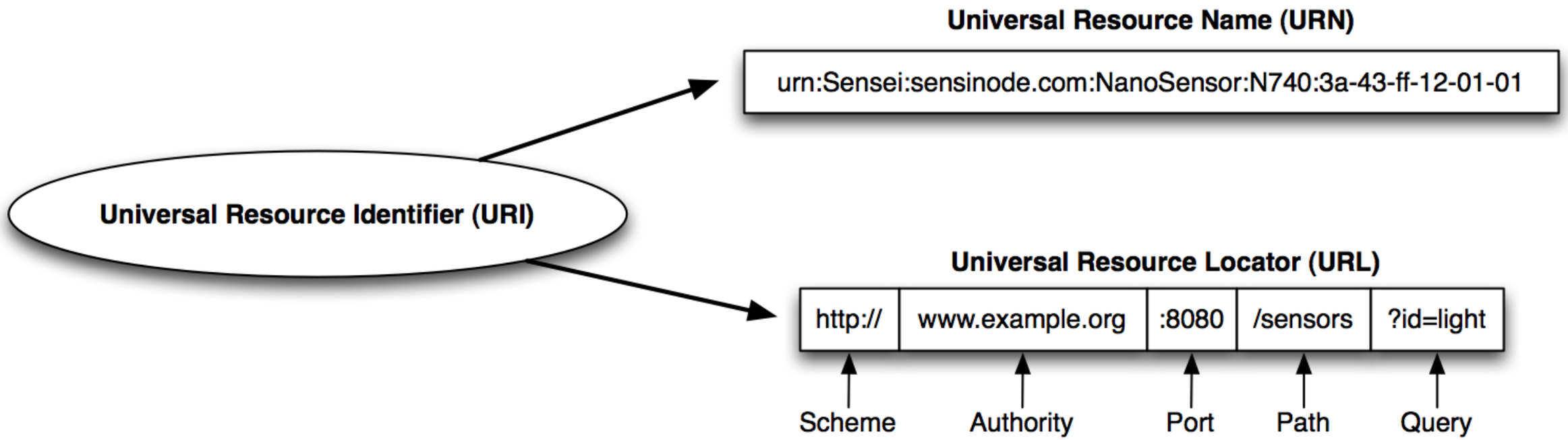
What are Web Services?

The Web Architecture

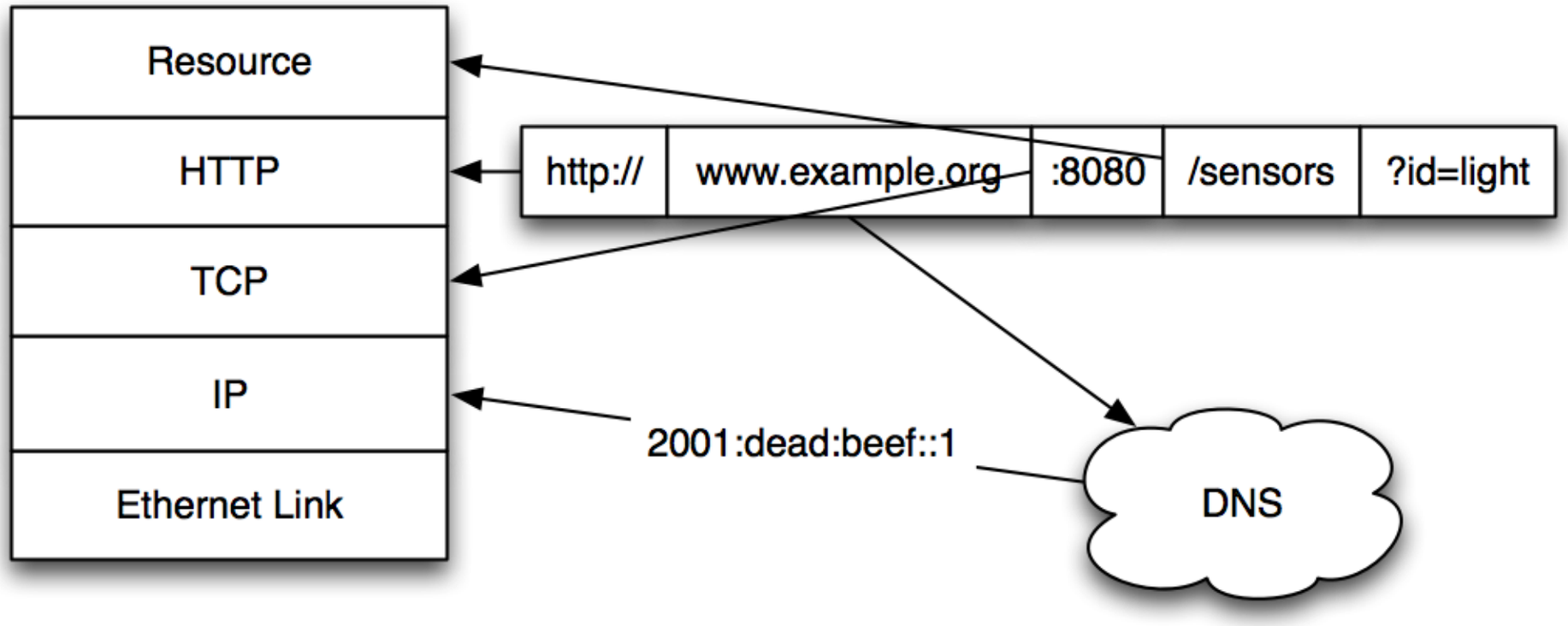
sensinode



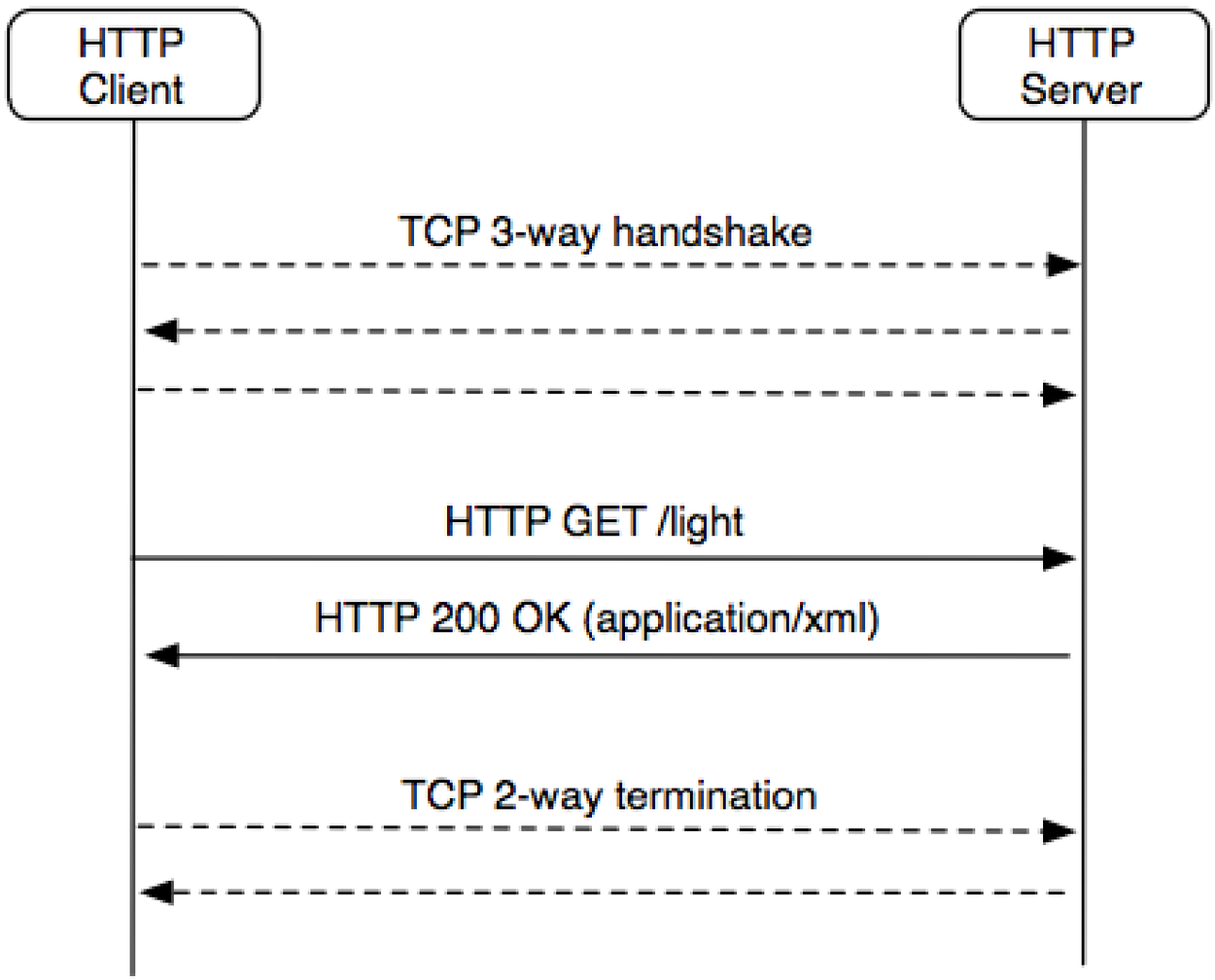
Web Naming


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

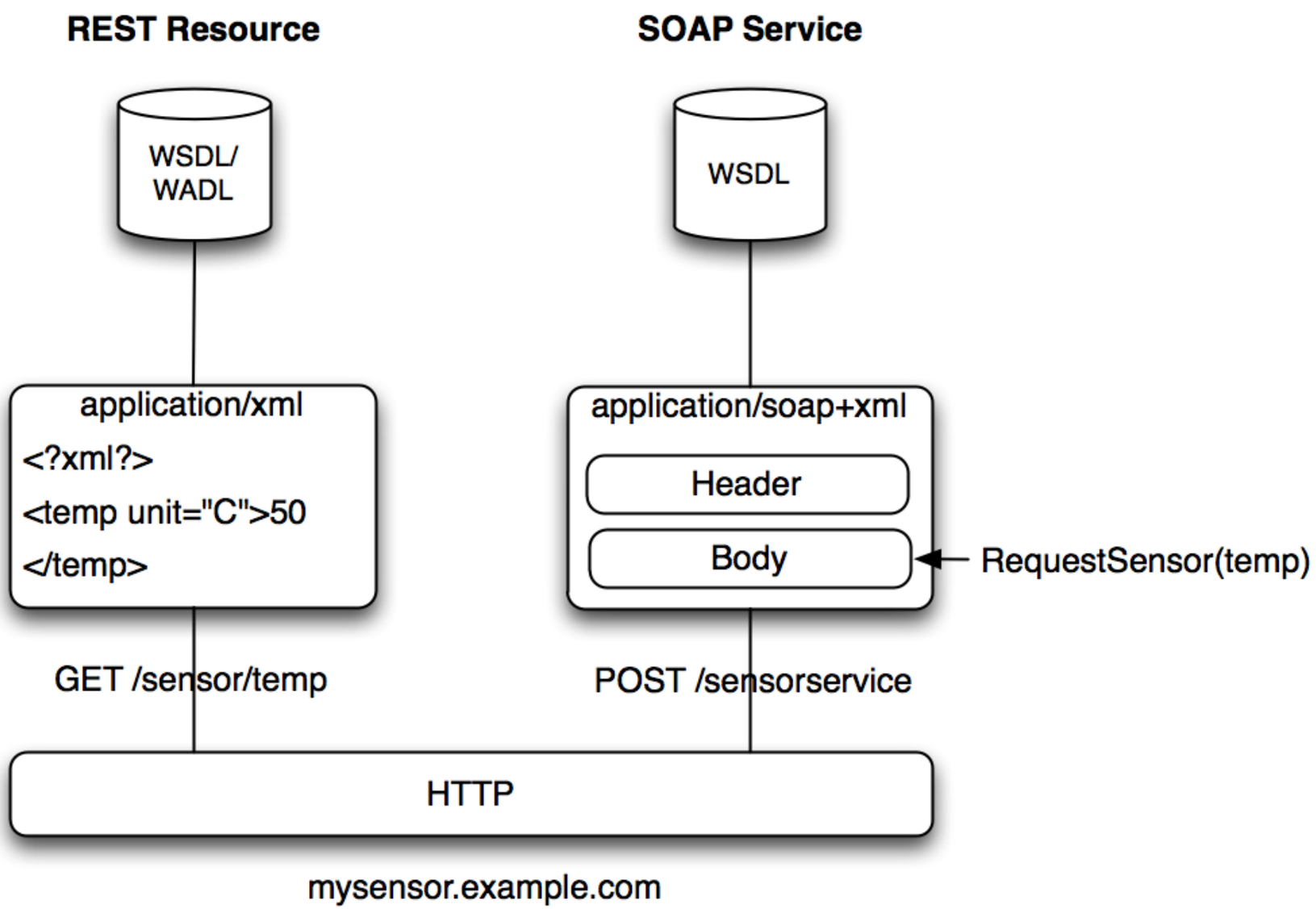


An HTTP Request

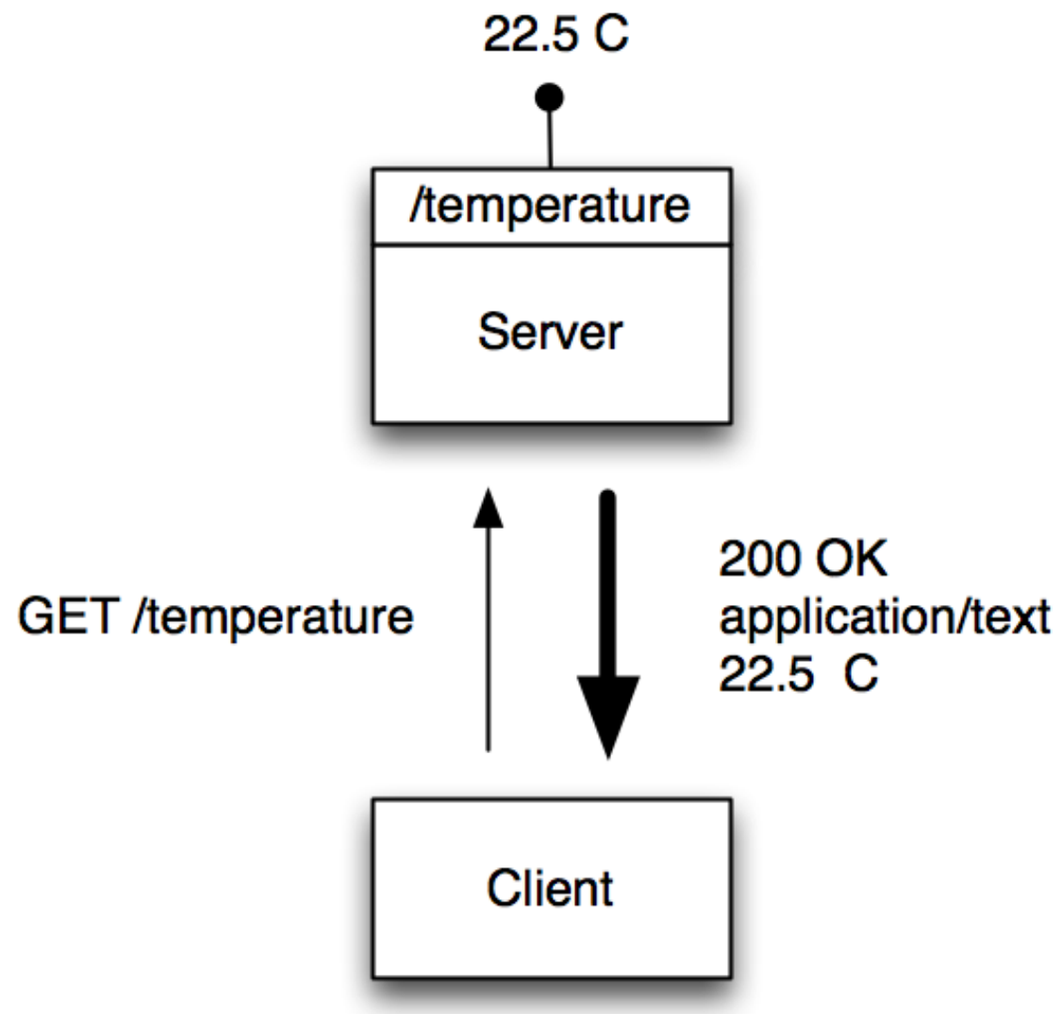


See RFC2616 - Hypertext Transfer Protocol v1.1

The Web Service Paradigm



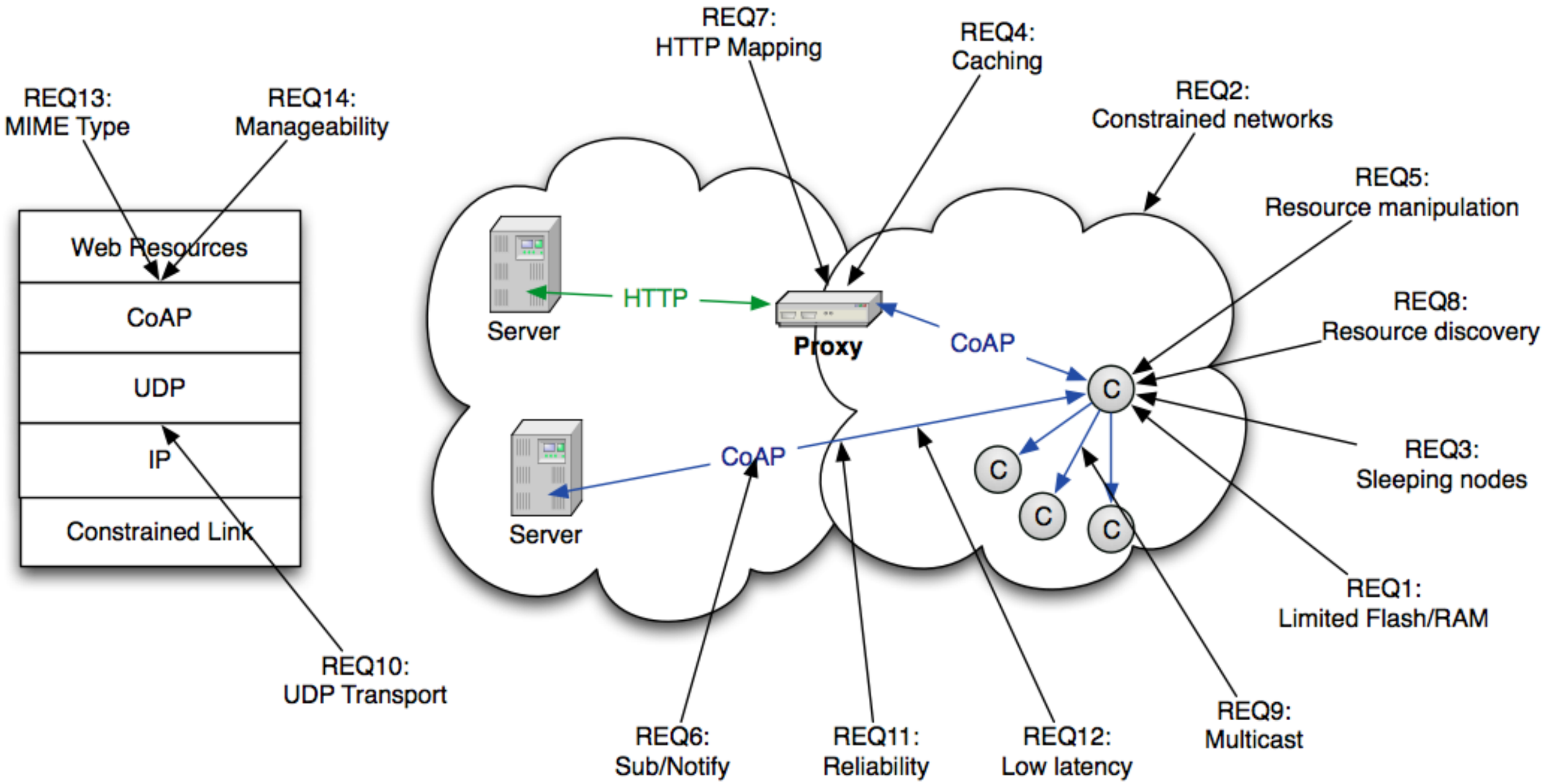
A REST Request



sensinode

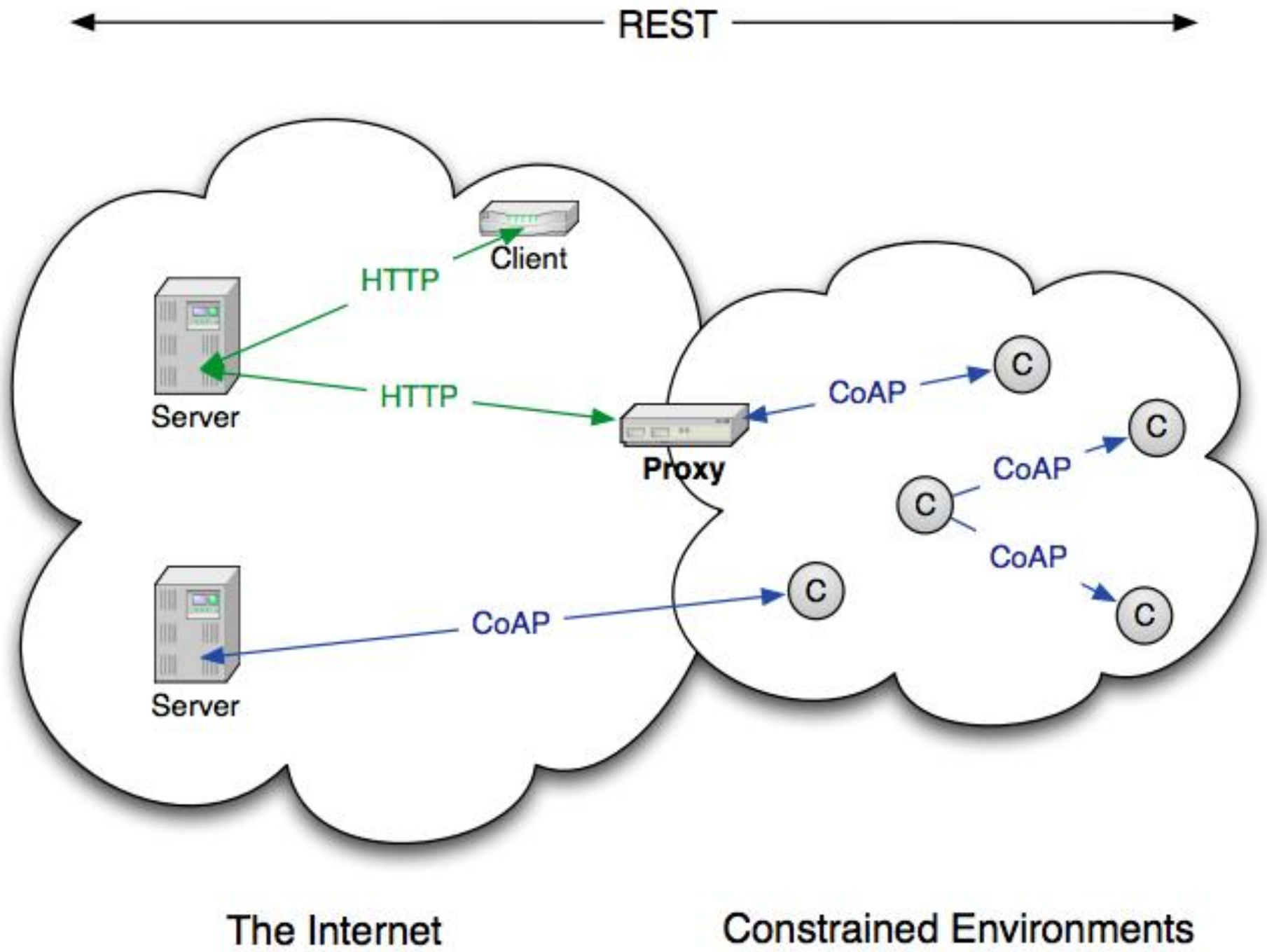
CoRE - Constrained RESTful Environments

CoRE Requirements



See draft-shelby-core-coap-req

The CoRE Architecture



The Constrained Application Protocol

sensinode

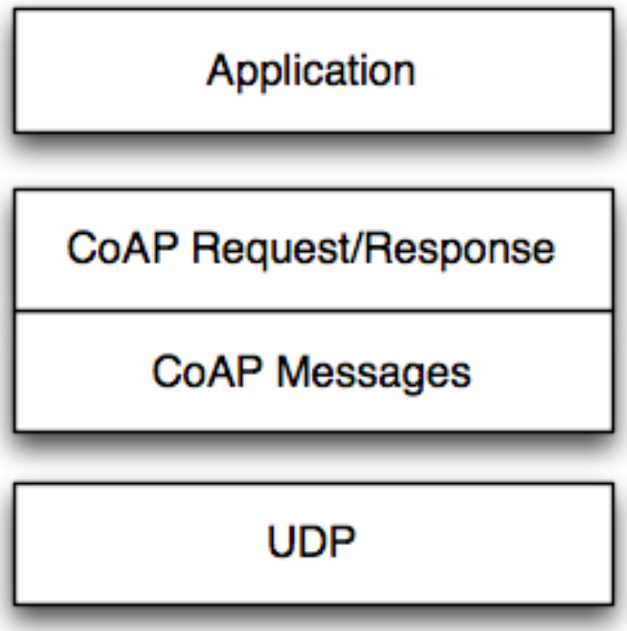
- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- Small, simple header < 10 bytes
- Subset of MIME types and HTTP-compatible response codes
- Optional observation, block transfer and discovery

What CoAP is (and is not)

- CoAP is
  - ✓A RESTful protocol
  - ✓Both synchronous and asynchronous
  - ✓For constrained devices and networks
  - ✓Specialized for M2M applications
  - ✓Easy to proxy to/from HTTP
- CoAP is not
  - ✓A replacement for HTTP
  - ✓General HTTP compression
  - ✓Separate from the web

The Transaction Model

- Transport
  - ✓ CoAP is defined for UDP
- Messaging
  - ✓ Simple message exchange between end-points
  - ✓ CON, NON, ACK, RST
- REST
  - ✓ Request/Response piggybacked on messages
  - ✓ Method, Response Code and Options (URI, content-type etc.)



Message Header

Ver - Version (1)

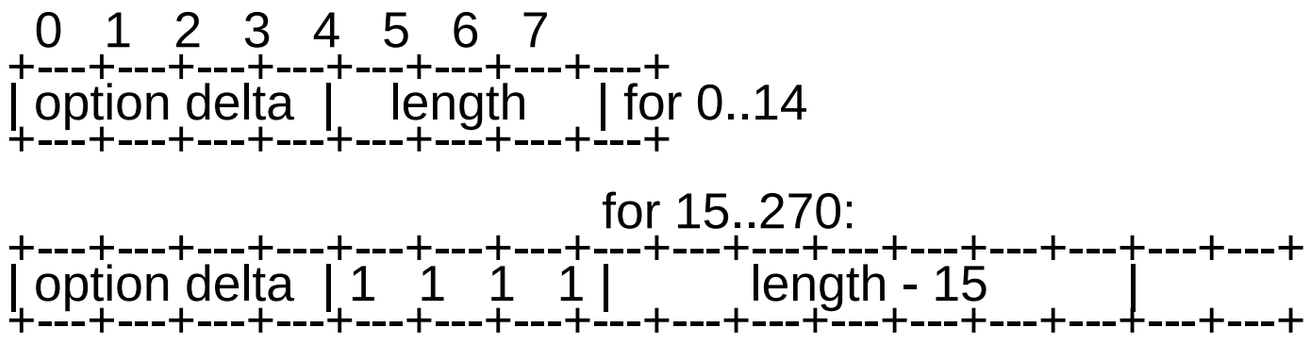
T - Transaction Type (Confirmable, Non-Confirmable, Acknowledgement, Reset)

OC - Option Count, number of options after this header

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

Message ID - Identifier for matching responses

Option Header



Option Delta - Difference between this option type and the previous

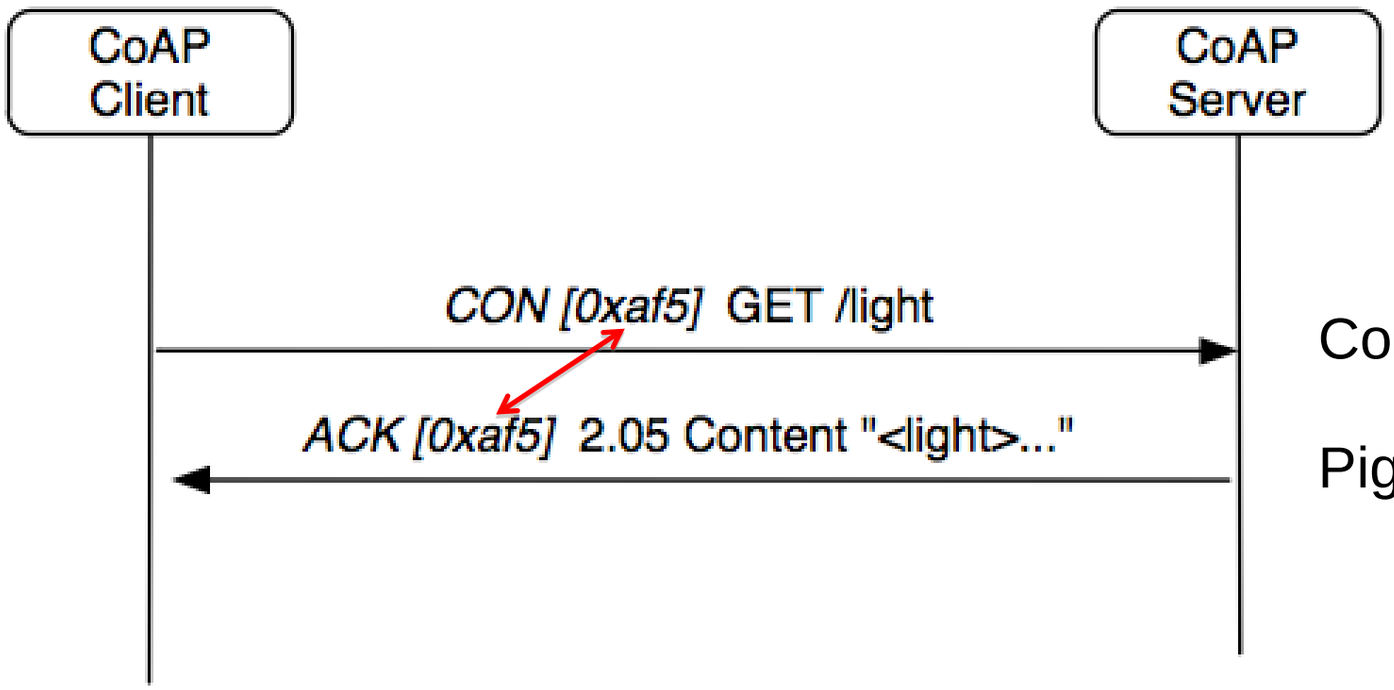
Length - Length of the option value (0-270)

Value - The value of Length bytes immediately follows Length

Options

No.	C/E	Name	Format	Length	Default
1	Critical	Content-Type	uint	0-2 B	(none)
2	Elective	Max-Age	uint	0-4 B	60
3	Critical	Proxy-Uri	string	1-270 B	(none)
4	Elective	ETag	opaque	1-8 B	(none)
5	Critical	Uri-Host	string	1-270 B	(see below)
6	Elective	Location-Path	string	1-270 B	(none)
7	Critical	Uri-Port	uint	0-2 B	(see below)
8	Elective	Location-Query	string	1-270 B	(none)
9	Critical	Uri-Path	string	1-270 B	(none)
11	Critical	Token	opaque	1-8 B	(empty)
12	Elective	Accept	uint	0-2 B	(none)
13	Critical	If-Match	opaque	0-8 B	(none)
15	Critical	Uri-Query	string	1-270 B	(none)
21	Critical	If-None-Match	(none)	0 B	(none)

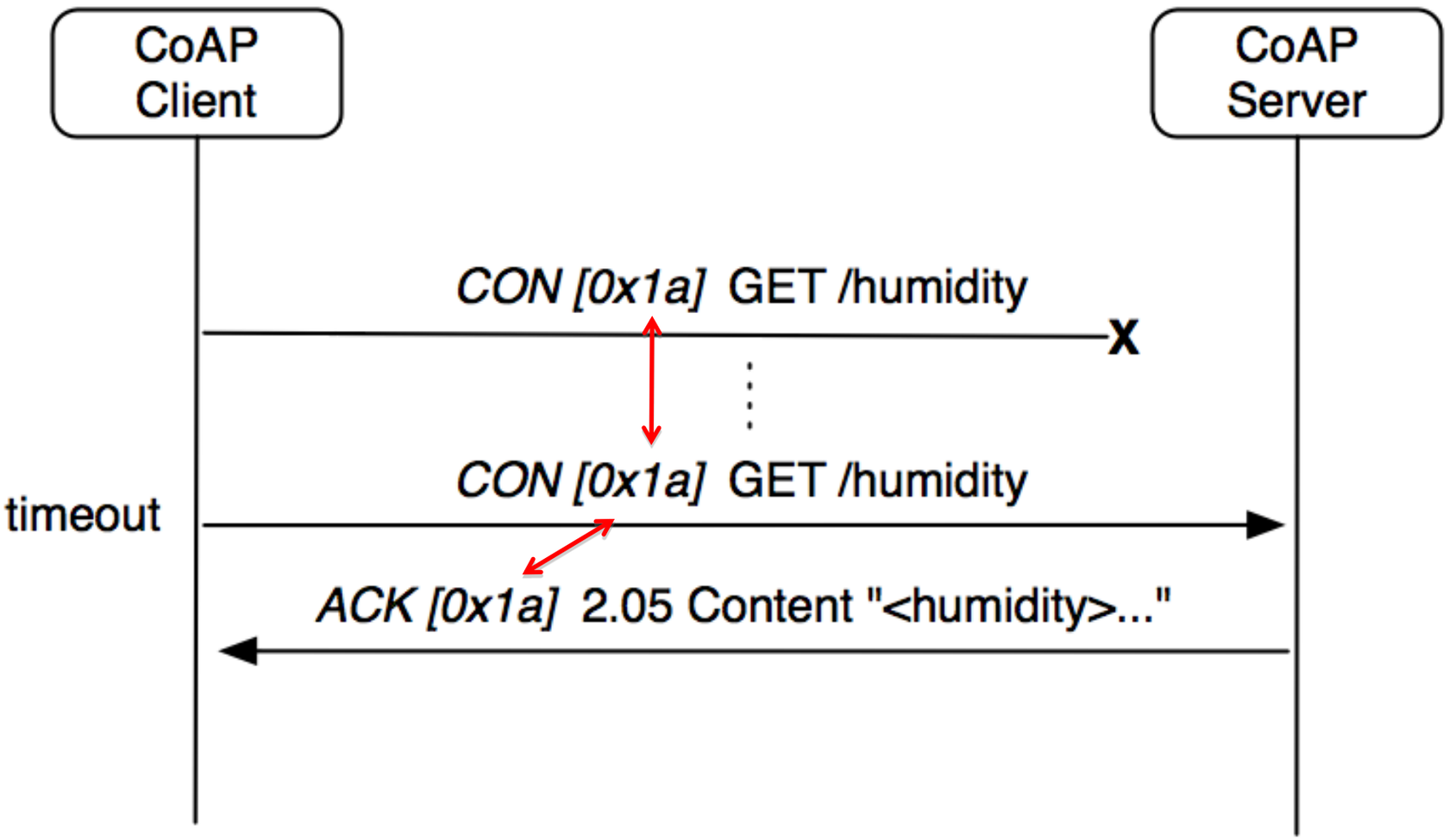
Request Example



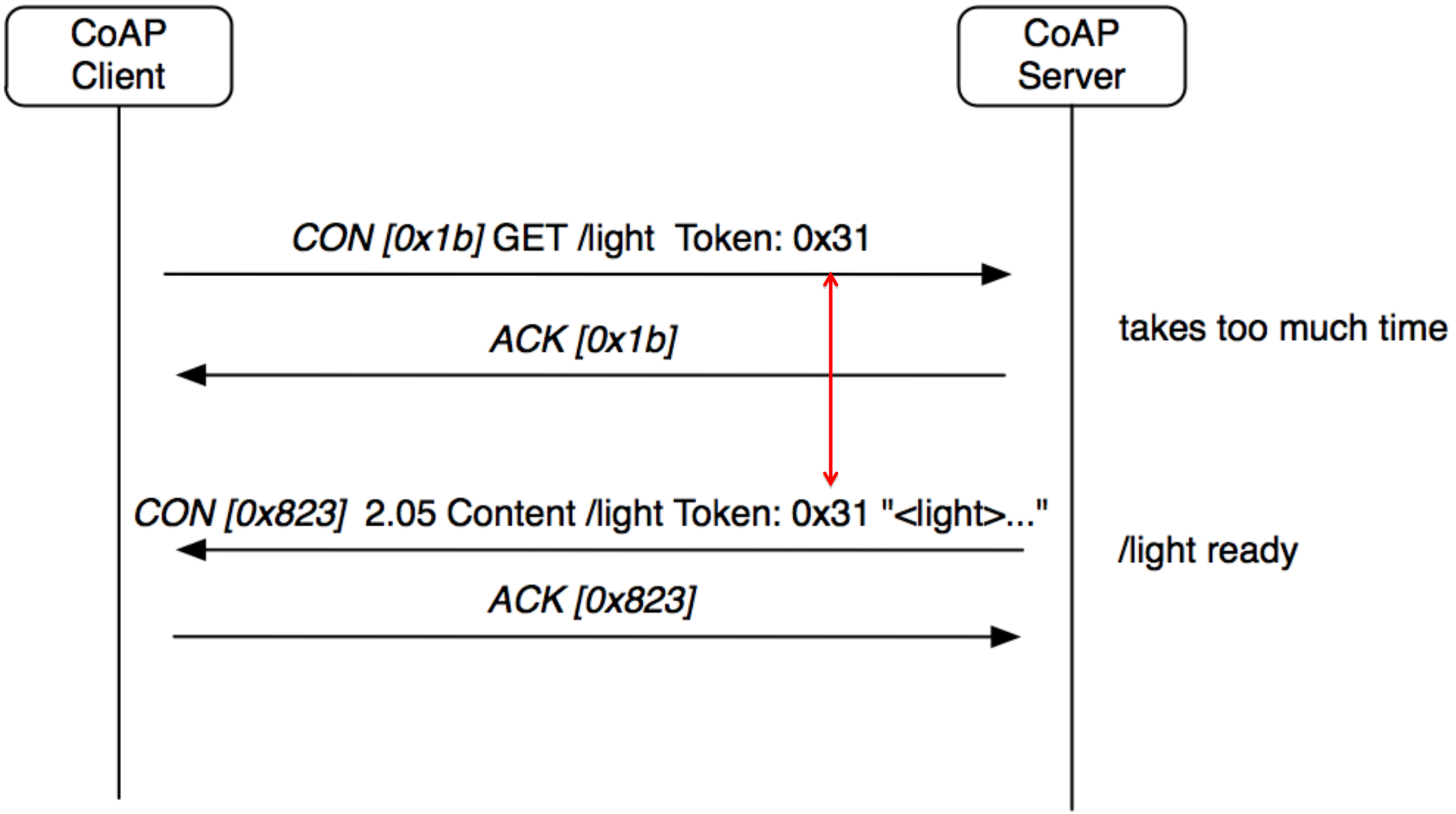
Confirmable Request

Piggy-backed Response

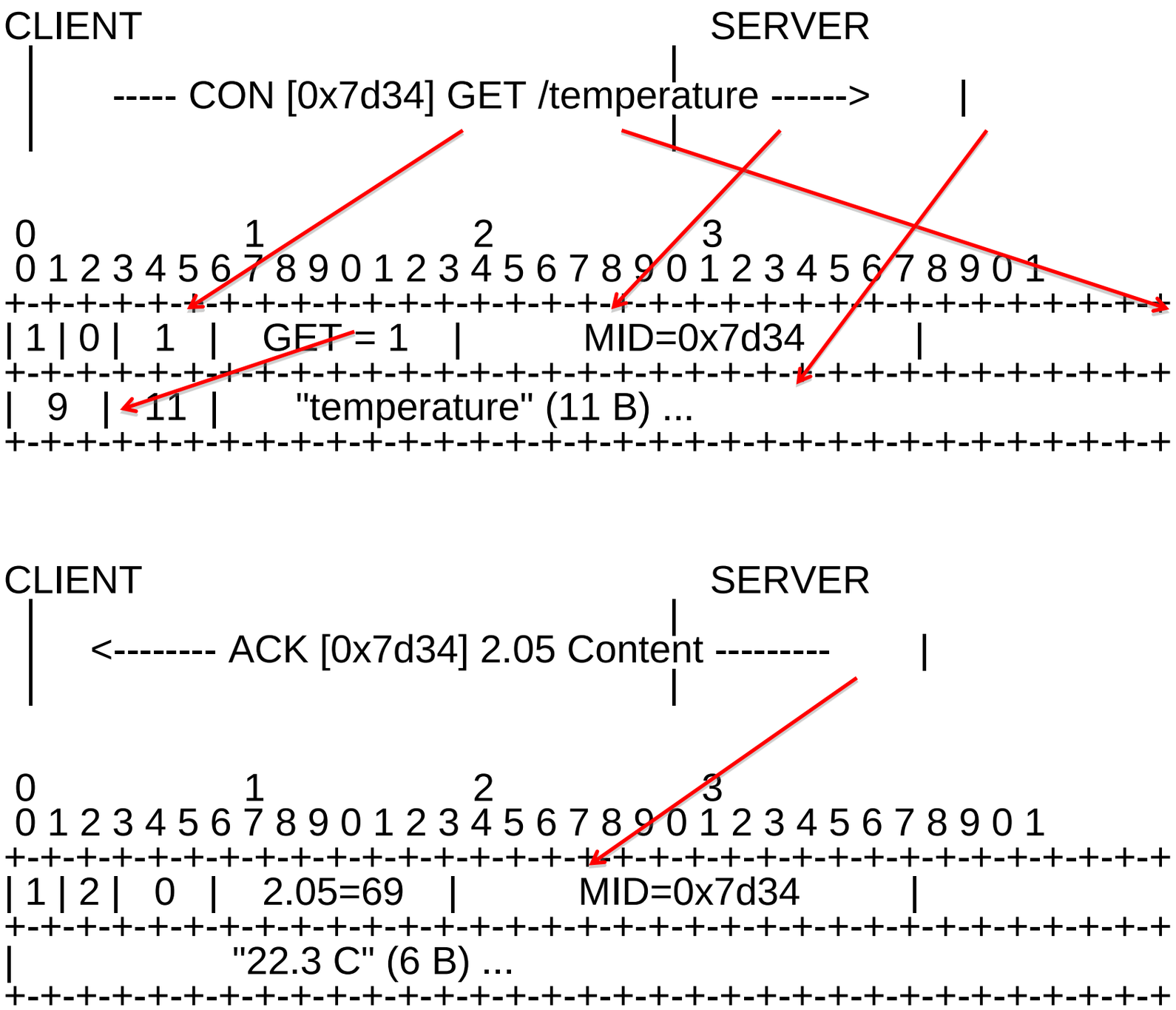
Dealing with Packet Loss



Normal Response



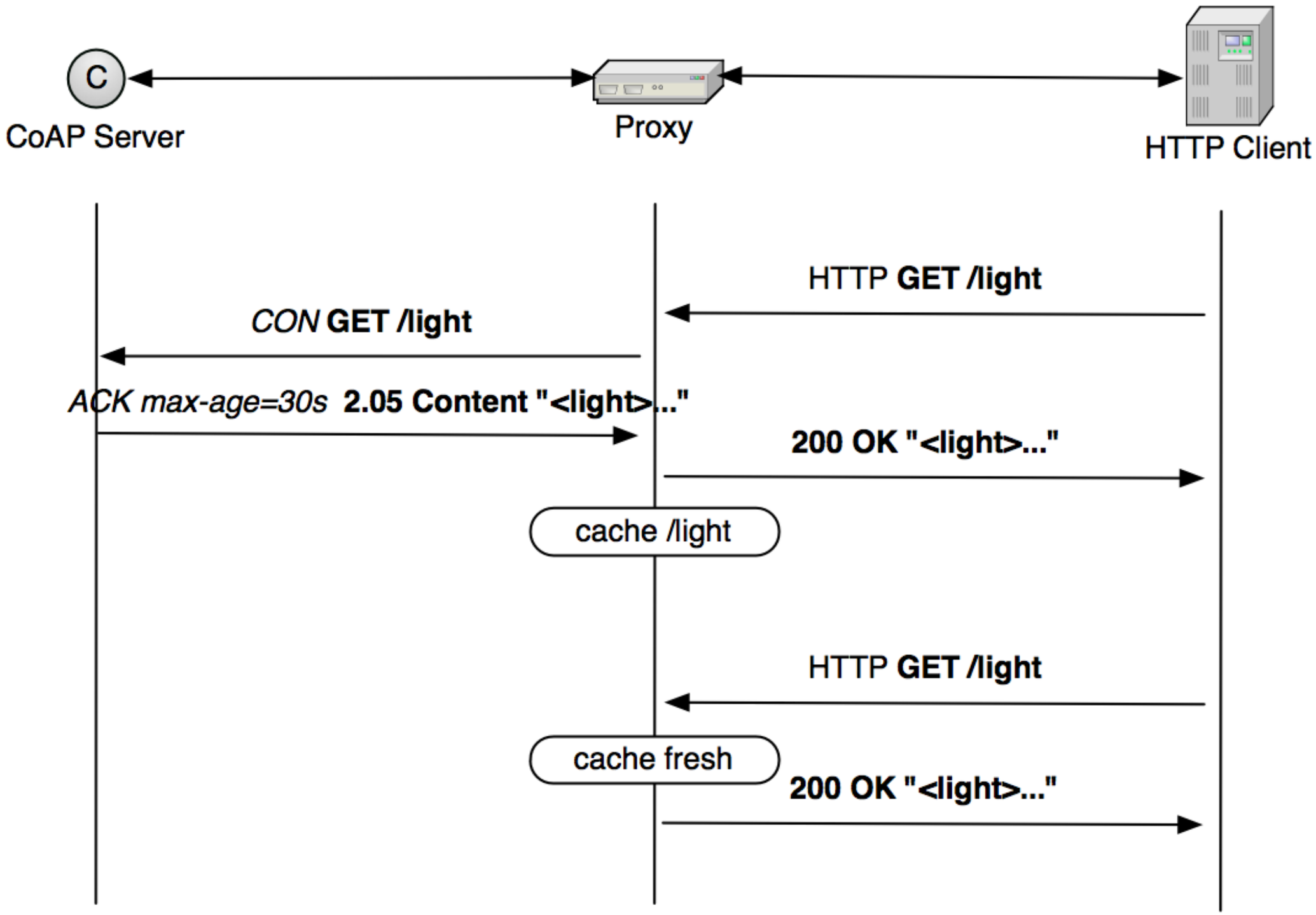
Bits and bytes...



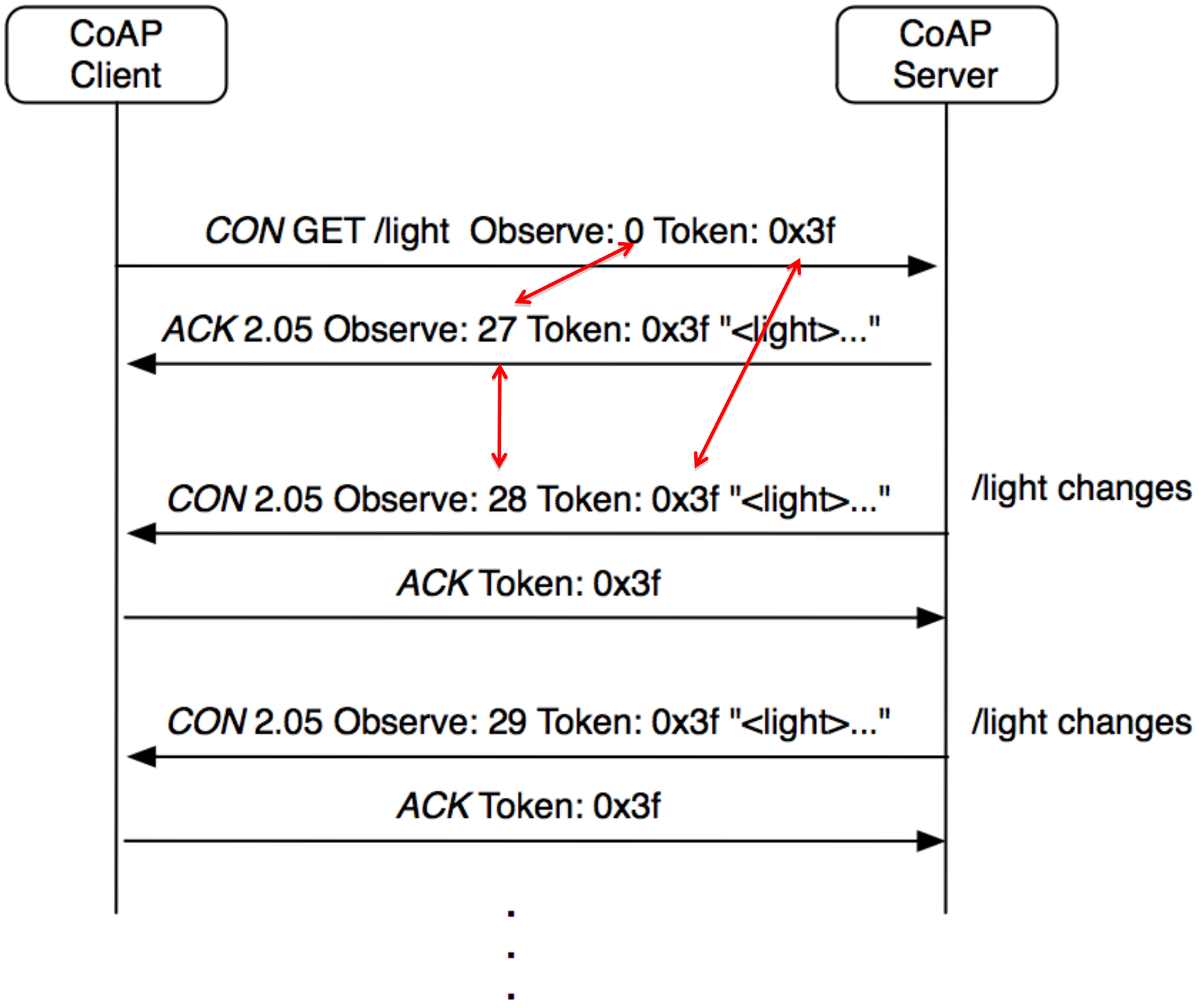
Caching

- CoAP includes a simple caching model
  - ✓ Cacheability determined by response code
- 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 sleeping node,
  - ✓ and to reduce network load

Proxying and caching

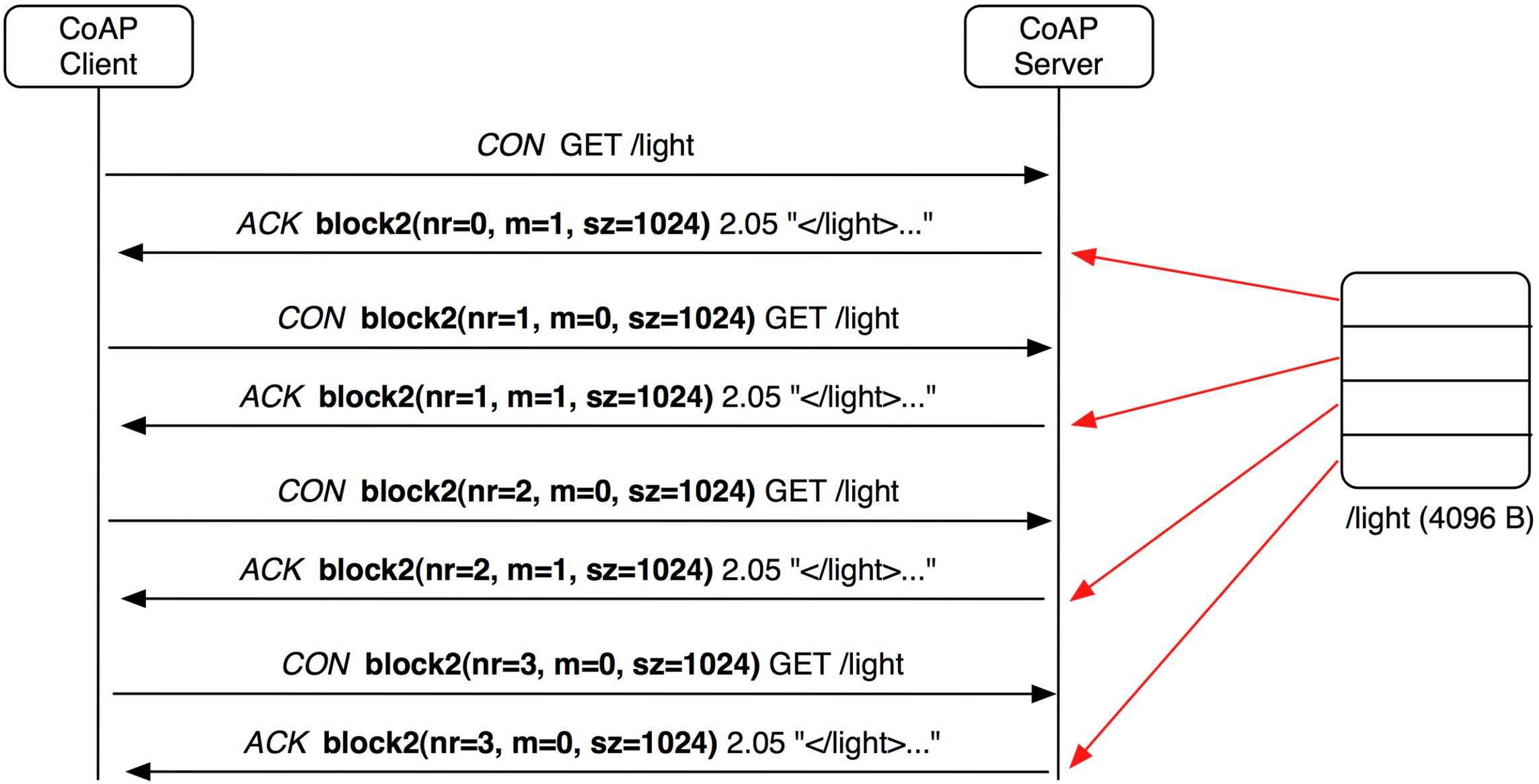


Observation



See draft-ietf-core-observe

Block transfer



See draft-ietf-core-block

sensinode

Web Linking

What is Web Linking?

- Links have been around a long time
- Web Linking formalizes links with defined relations, **typed links**
  - ✓ HTML and Atom have allowed links for a long time
- 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

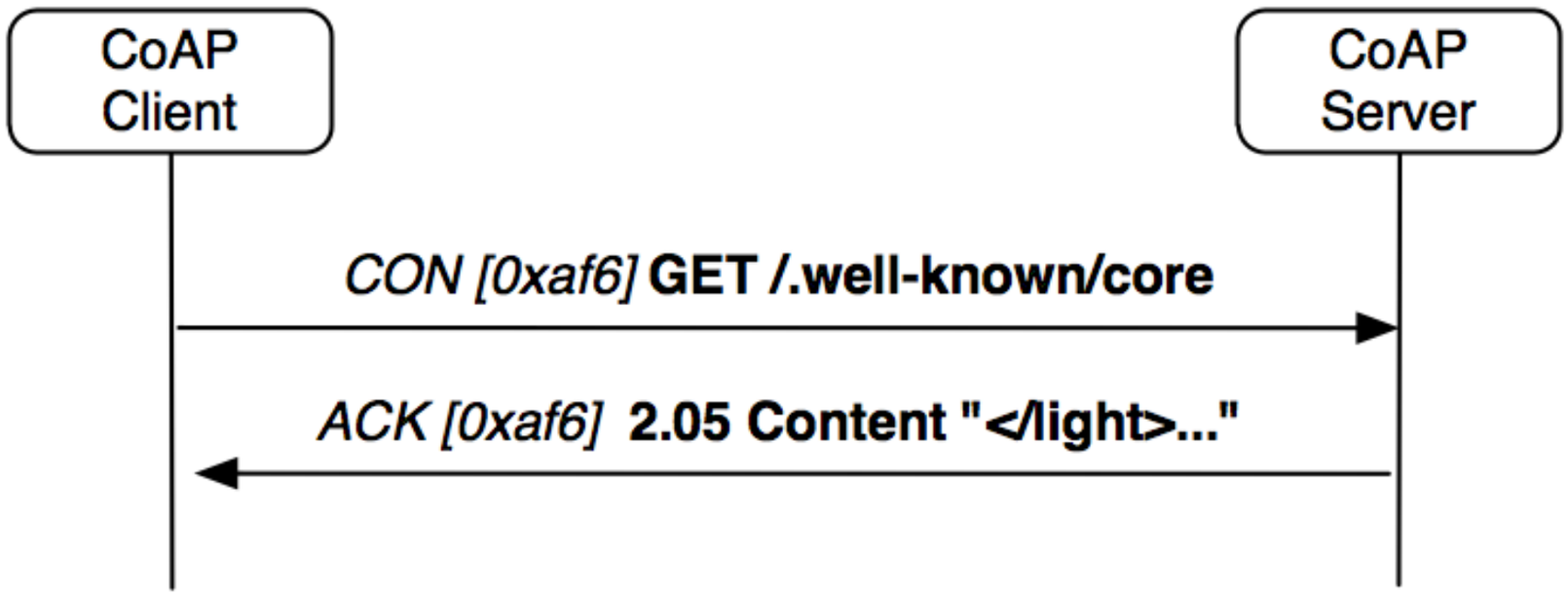
- Service Discovery
  - ✓ What services are available in the first place?
  - ✓ Goal of finding the IP address, port and protocol
  - ✓ Usually performed by e.g. DNS-SD when DNS is available
- Resource Discovery
  - ✓ What are the Web resources I am interested in?
  - ✓ Goal of finding URIs
  - ✓ Performed using Web Linking or some REST interface
    - ✓ The EU FP7 SENSEI project did work on this
    - ✓ CoRE Link Format is designed to enable resource discovery

CoRE Link Format

- CoRE Link Format 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 CoRE Link Format
  - ✓ 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.

See draft-ietf-core-link-format-05

CoRE Resource Discovery



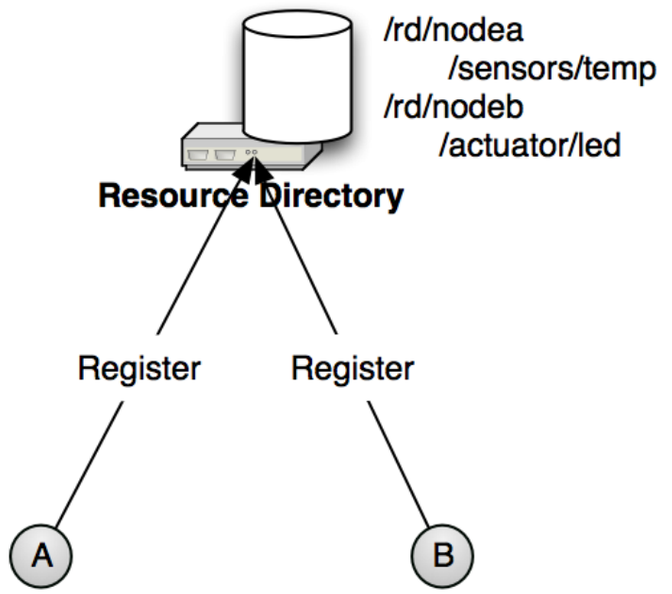
</light>;rt="Illuminance";ct=0,
</s/maastr.xml>;title="Maastricht weather";ct=1,
</s/maastr/temp>;title="Temperature in Maastrich";ct=1,
</s/oulu.xml>;title="Oulu weather";ct=1,
</s/oulu/temp>;title="Temperature in Oulu";ct=1,
</s/temp>;rt="Temperature";ct=0

Semantic Soup

- 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-06]
- What can we make universal?
- What should be market specific?
- How do we enable innovation?

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



/sensor/temp /actuator/led