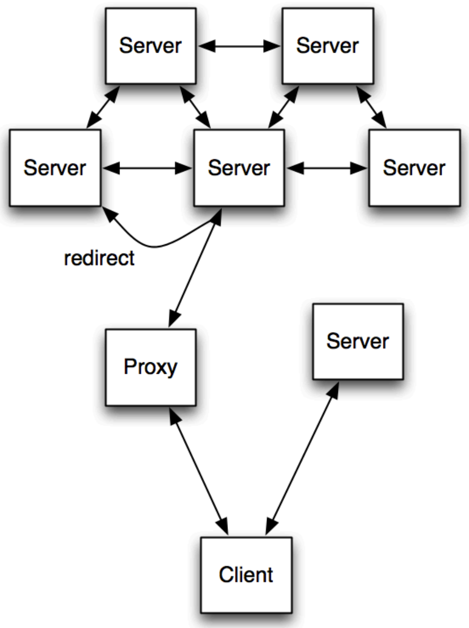
Constrained Application Protocol (CoAP)

Background

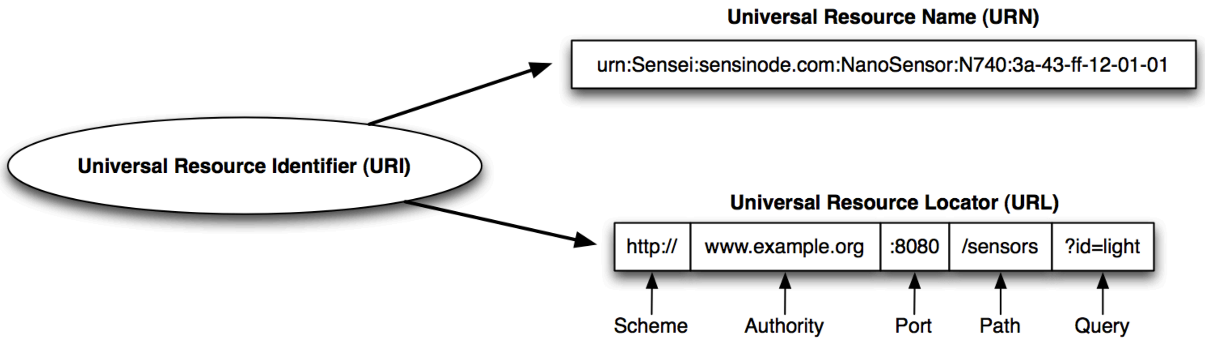
- GOAL: to enable web-based services in constrained wireless networks
  - 8 bit micro-controllers
  - limited memory
  - low-power networks
- Problem: WEB solution are hardly applicable
- Solution: re-design web-based services for constrained networks -> COAP

How Does the Web Work?

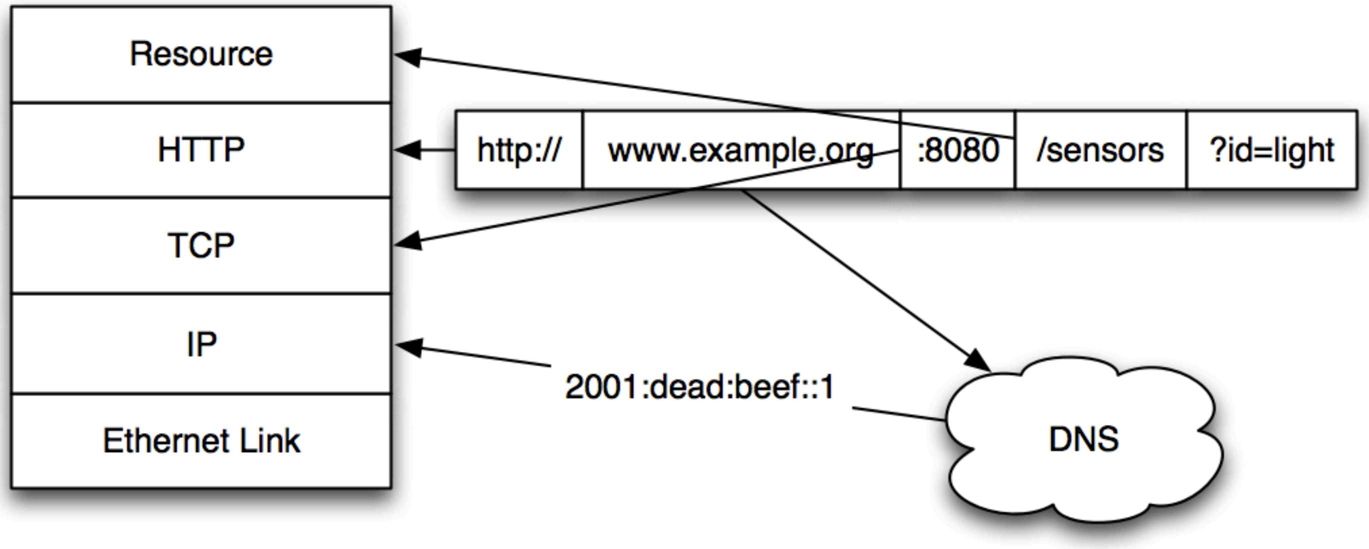
- Resources in the Web are:
  - managed by servers
  - identified by URIs
  - accessed synchronously by clients through request/ response paradigms
- In a word, Representational State Transfer (REST)



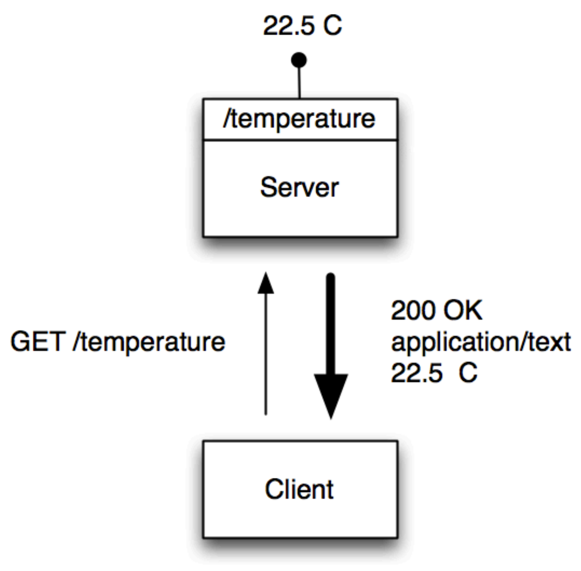
URL Resolution

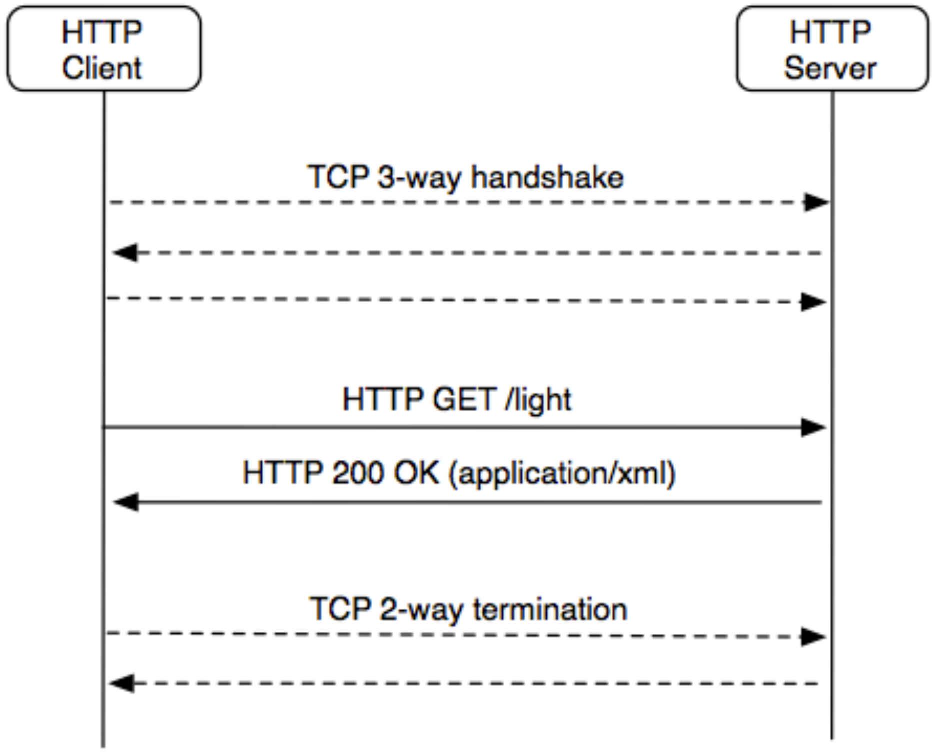


Universal Resource Name (URN)
urn:Sensei:sensinode.com:NanoSensor:N740:3a-43-ff-12-01-01
Universal Resource Identifier (URI)
Universal Resource Locator (URL)
http://
www.example.org
:8080
/sensors
?id=light
Scheme
Authority
Port
Path
Query



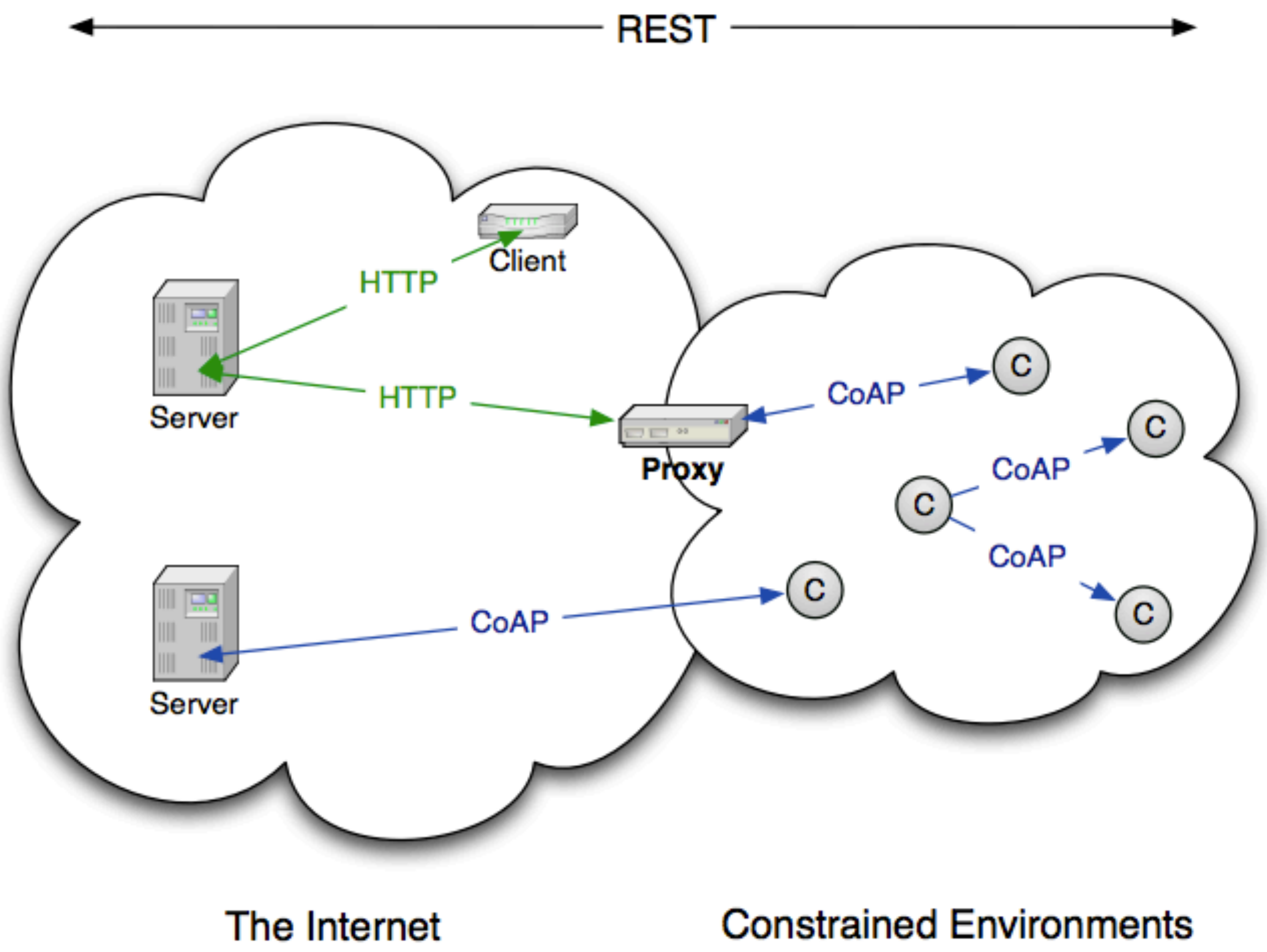
Request/Response Transaction



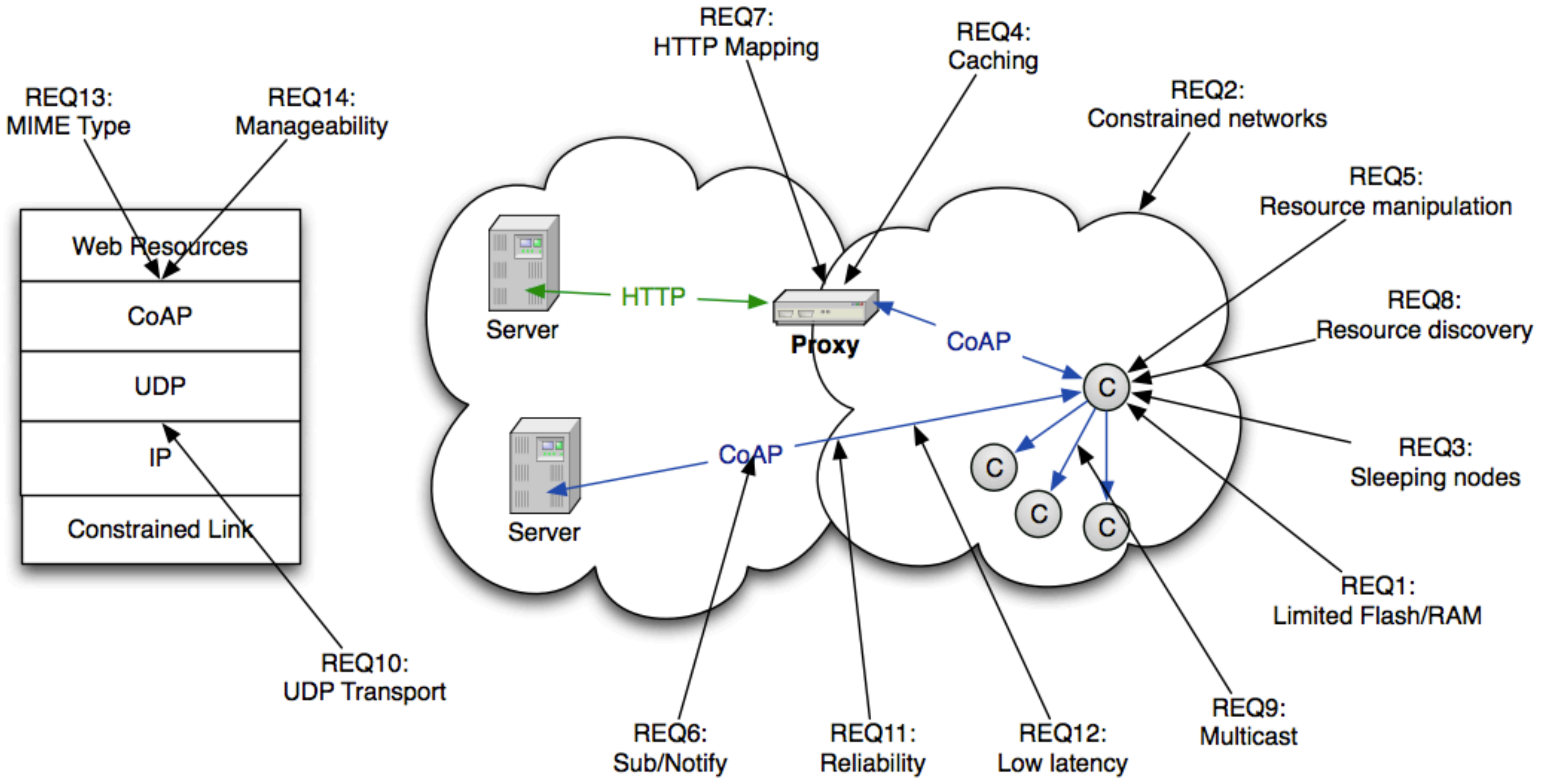


- Other common HTTP methods: PUT, POST, DELETE

The CoAP Architecture



CoAP Design Requirements



See draft-shelby-core-coap-req

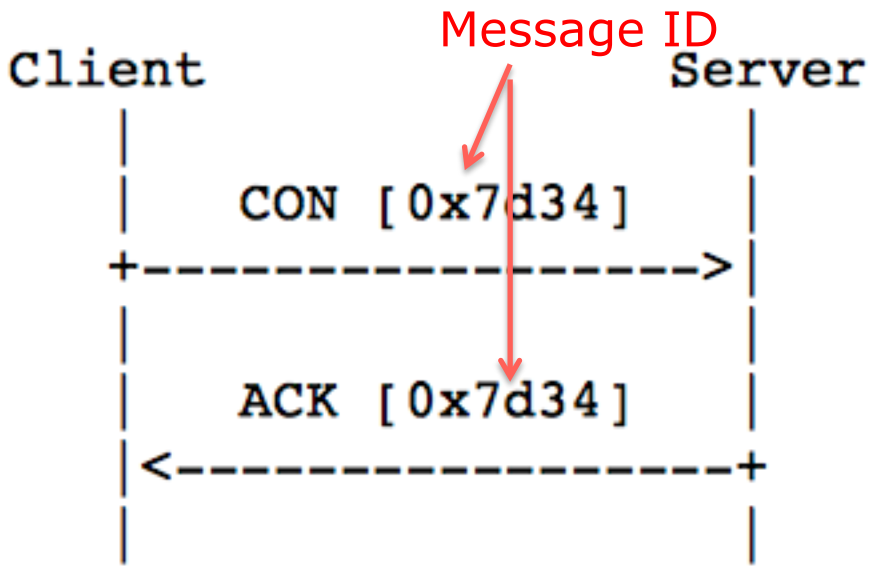
CoAP At a Glance

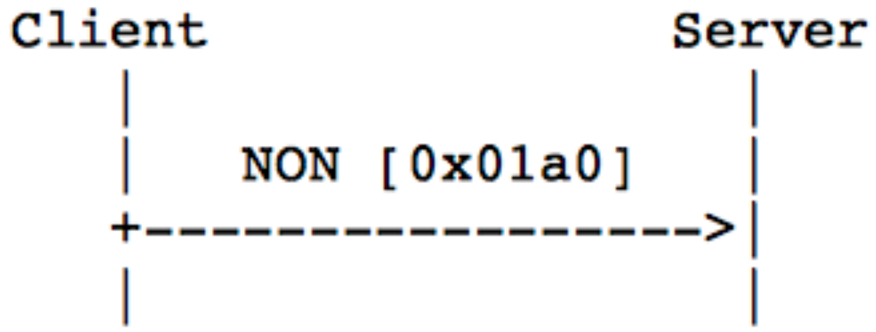
- Embedded web transfer protocol (coap://)
- Asynchronous transaction model
- UDP binding with reliability and multicast support
- GET, POST, PUT, DELETE methods
- URI support
- 4 byte header
- Subset of MIME types and HTTP response codes
- Built-in discovery
- Optional observation and block transfer

COAP Messaging Basics

- Transport:
  - (mainly) UDP binding
- Message Exchange between Endpoints
  - Messages with 4 bytes header (shared by request and responses) containing a message ID (16 bits)
  - Reliable exchange through Confirmable Messages which must be acknowledged (through ACK or Reset Messages). Simple Stop-and-Wait retransmission with exponential back-off.
  - Unreliable exchange through Non-Confirmable Message
  - Duplicate detection for both confirmable and non-confirmable messages (through message ID)

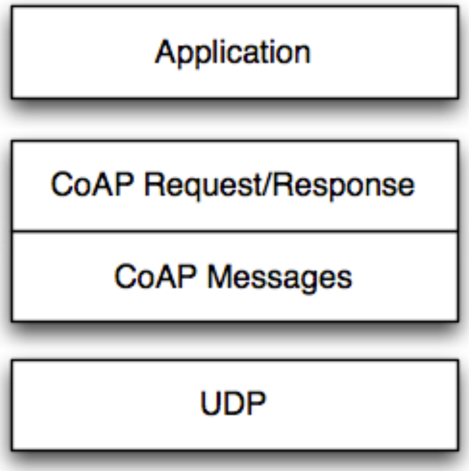
COAP Messaging



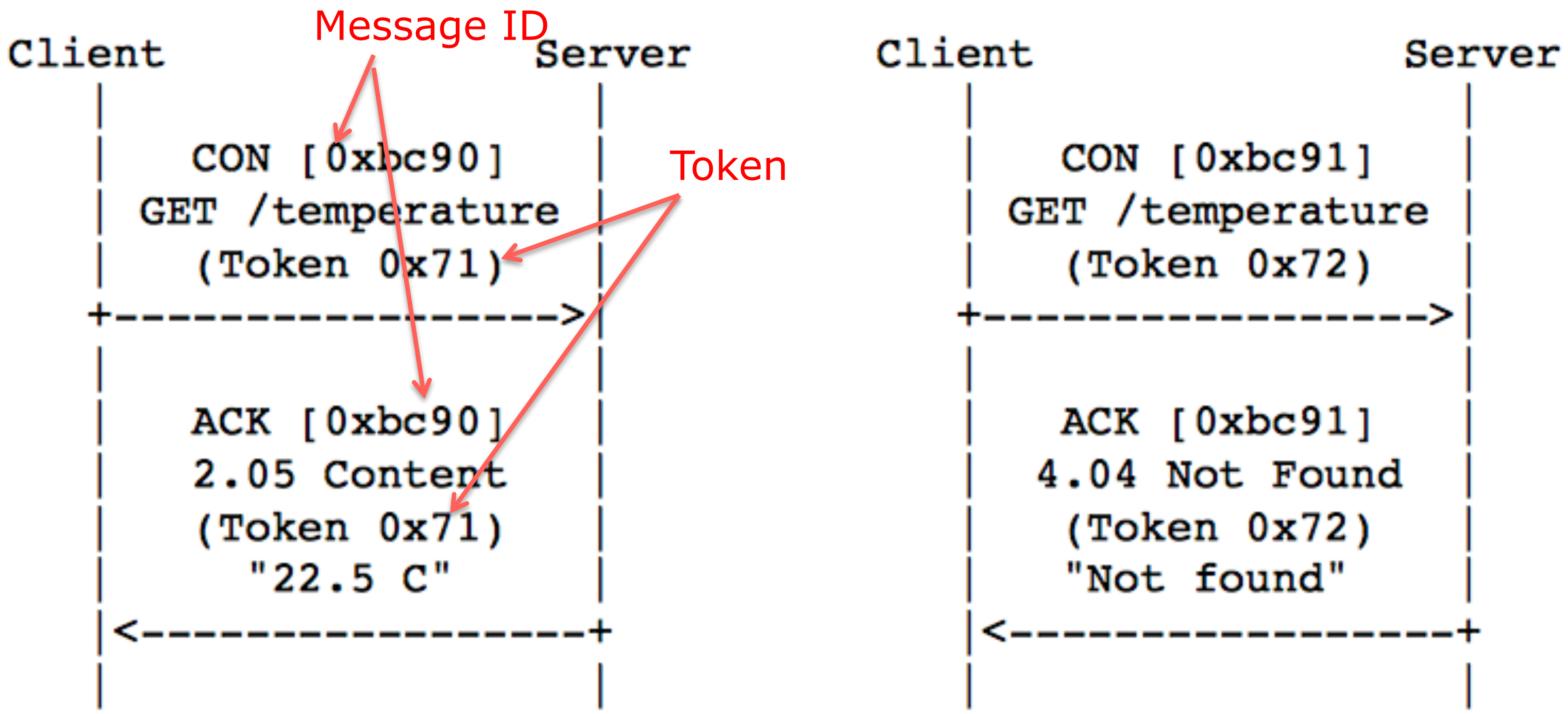


COAP Message Semantics

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

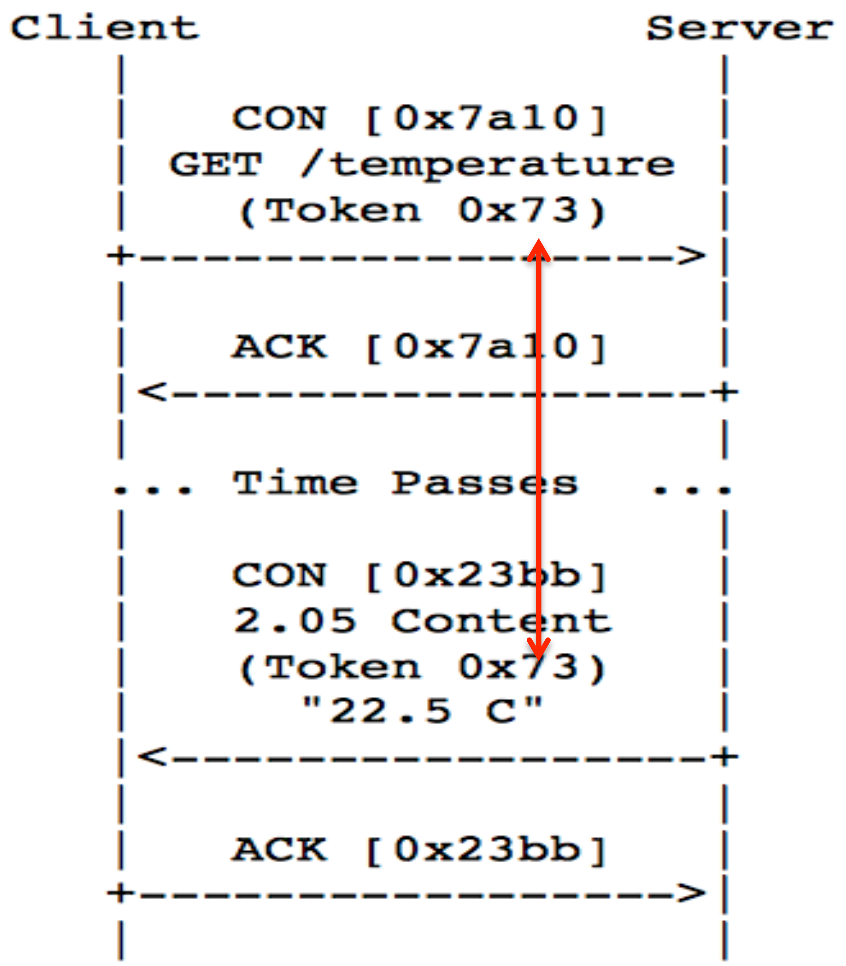


COAP Request/Response Examples



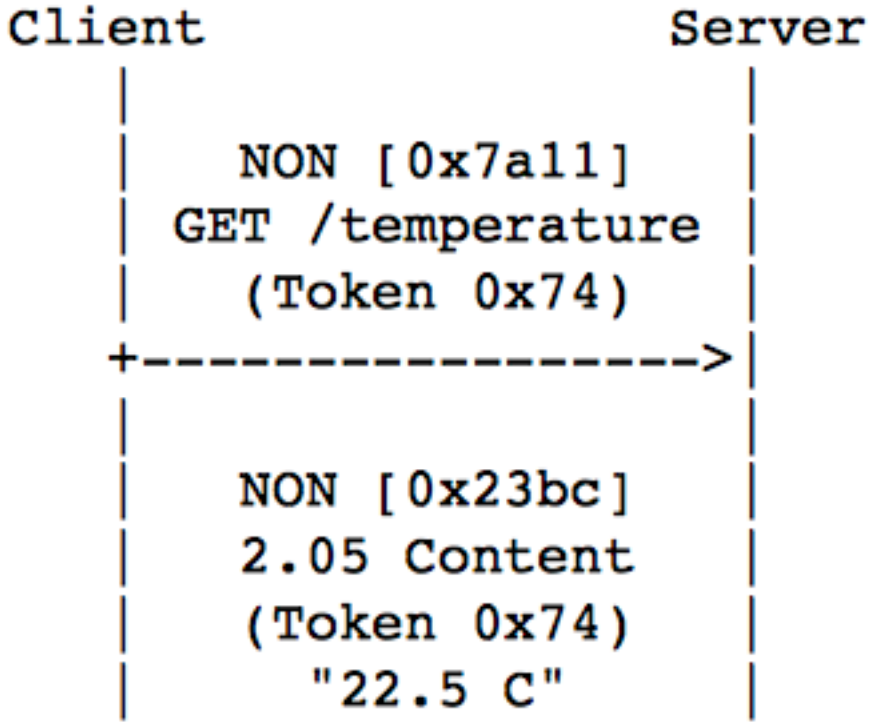
```
Client                    Server          Client                    Server
  |                          |              |                          |
  |     CON [0xbc90]         |              |     CON [0xbc91]         |
  |  GET /temperature        |              |  GET /temperature        |
  |    (Token 0x71)          |              |    (Token 0x72)          |
  +------------------------->|              +------------------------->|
  |                          |              |                          |
  |     ACK [0xbc90]         |              |     ACK [0xbc91]         |
  |    2.05 Content          |              |   4.04 Not Found         |
  |    (Token 0x71)          |              |    (Token 0x72)          |
  |      "22.5 C"            |              |     "Not found"          |
  |<-------------------------+              |<-------------------------+
  |                          |              |                          |
```

COAP: Separate Response



```
Client                Server
   |                     |
   |   CON [0x7a10]      |
   | GET /temperature    |
   |   (Token 0x73)      |
   +-------------------->|
   |                     |
   |   ACK [0x7a10]      |
   |<--------------------+
   |                     |
   ... Time Passes    ...
   |                     |
   |   CON [0x23bb]      |
   |   2.05 Content      |
   |   (Token 0x73)      |
   |     "22.5 C"        |
   |<--------------------+
   |                     |
   |   ACK [0x23bb]      |
   +-------------------->|
   |                     |
```

COAP: Non-confirmable Request



```
Client              Server
   |                  |
   |  NON [0x7a11]    |
   | GET /temperature |
   |  (Token 0x74)    |
   +----------------->|
   |                  |
   |  NON [0x23bc]    |
   |  2.05 Content    |
   |  (Token 0x74)    |
   |    "22.5 C"      |
```

Message Header (4 bytes)

```
  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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

```
  0   1   2   3   4   5   6   7
+---------------+---------------+
|               |               |
|  Option Delta | Option Length |   1 byte
|               |               |
+---------------+---------------+
\                               \
/         Option Delta          /   0-2 bytes
\          (extended)           \
+-------------------------------+
\                               \
/         Option Length         /   0-2 bytes
\          (extended)           \
+-------------------------------+
\                               \
/                               /
\                               \
/         Option Value          /   0 or more bytes
\                               \
/                               /
\                               \
+-------------------------------+
```

Option Delta - Difference between this option type and the previous

Length - Length of the option value

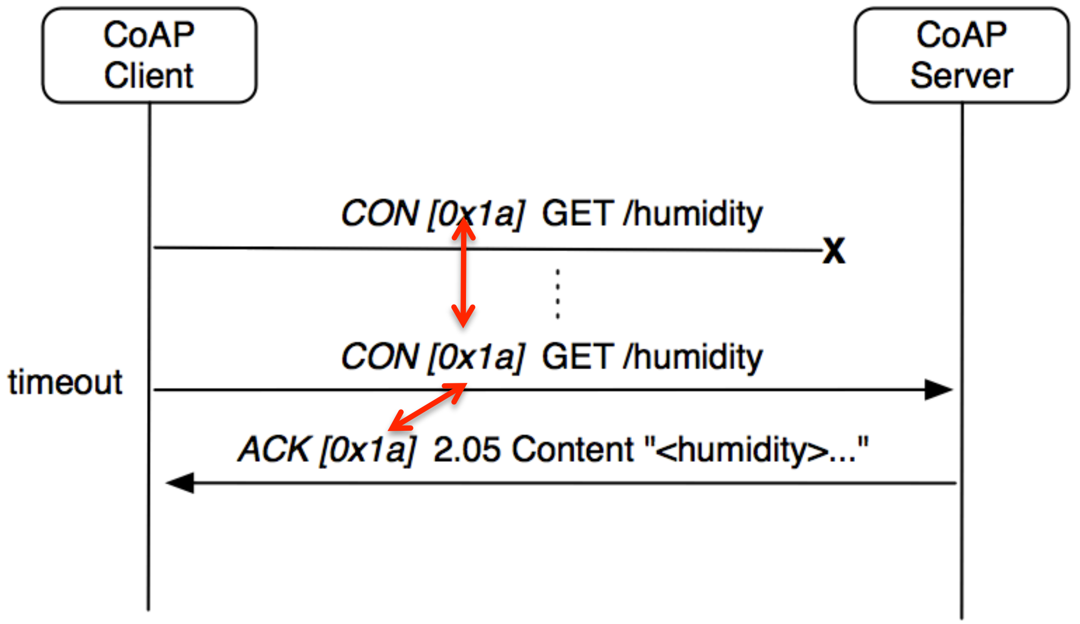
Value - The value of Length bytes immediately follows Length

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

Dealing with Packet Loss

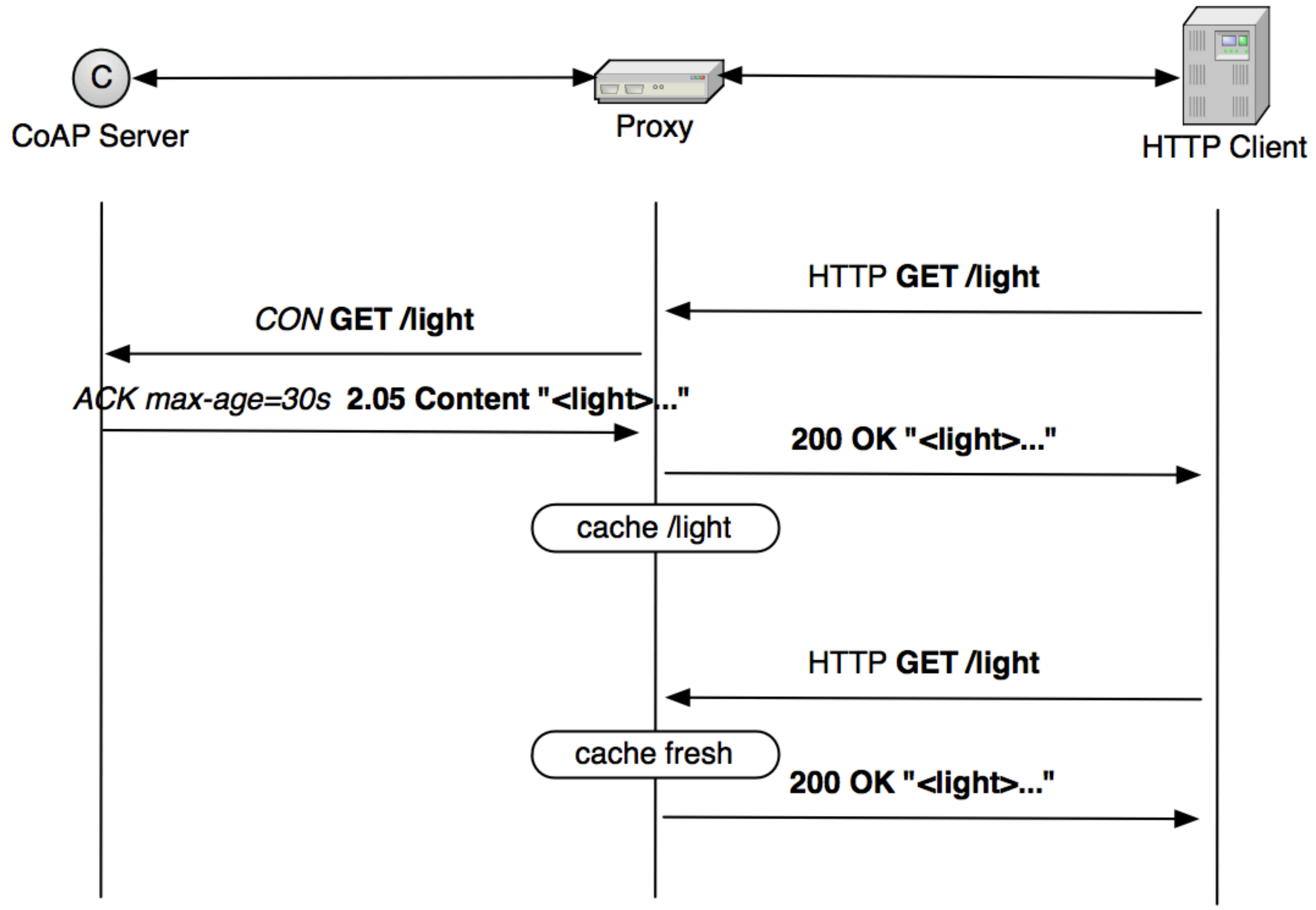


- Stop and Wait approach
- Repeat a request after a time-out in case ACK (or RST) is not coming back

Back-Off Details

- Initial time-out set to:
  - Rand [ACK_TIMEOUT, ACK_TIMEOUT * ACK_RANDOM_FACTOR] ([2s, 3s])
- When time-out expires and the transmission counter is less than MAX_RETRANSMIT (4)
  - retransmit
  - Increase transmission counter
  - double the time-out value
- The procedure is repeated until
  - A ACK is received
  - A RST message is received
  - the transmission counter exceeds MAX_RETRANSMIT
  - the total attempt duration exceeds MAX_TRANSMIT_WAIT (93s)

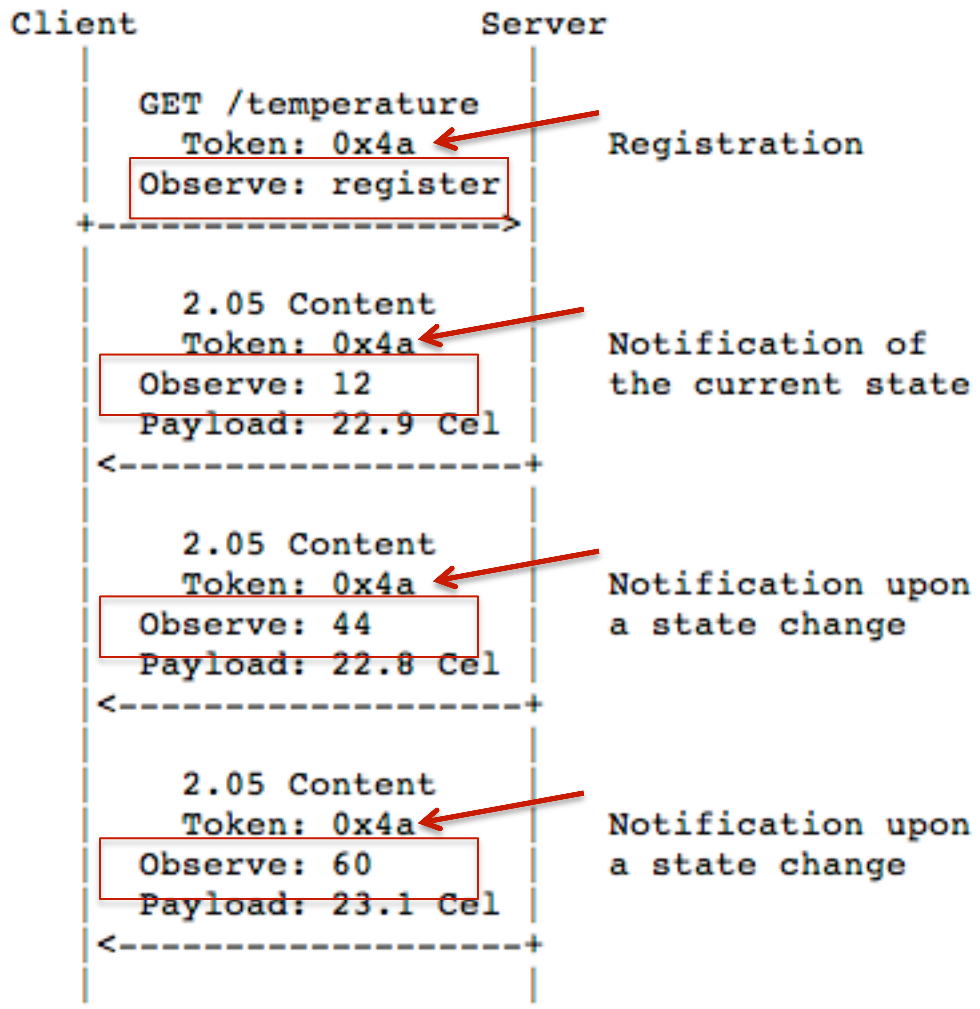
Proxying and caching



COAP Observation

- PROBLEM:
  - REST paradigm is often “PULL” type, that is, data is obtained by issuing an explicit request
  - Information/data in WSN is often periodic/ triggered (e.g., get me a temperature sample every 2 seconds or get me a warning if temperature goes below 5°C)
- SOLUTION: use Observation on COAP resources

Observation



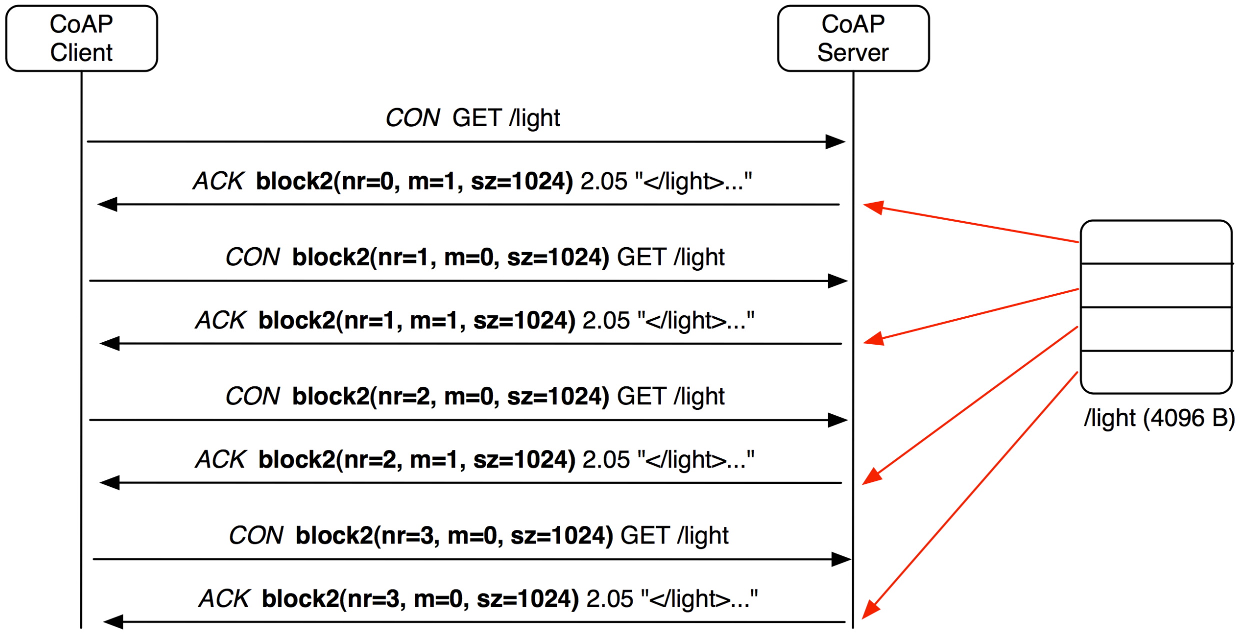
```
Client                  Server
   |                       |
   |  GET /temperature     |
   |    Token: 0x4a        |   Registration
   |  Observe: register    |
   +---------------------->|
   |                       |
   |    2.05 Content       |
   |    Token: 0x4a        |   Notification of
   |  Observe: 12          |   the current state
   |  Payload: 22.9 Cel    |
   |<----------------------+
   |                       |
   |    2.05 Content       |
   |    Token: 0x4a        |   Notification upon
   |  Observe: 44          |   a state change
   |  Payload: 22.8 Cel    |
   |<----------------------+
   |                       |
   |    2.05 Content       |
   |    Token: 0x4a        |   Notification upon
   |  Observe: 60          |   a state change
   |  Payload: 23.1 Cel    |
   |<----------------------+
   |                       |
```

See draft-ietf-core-observe

COAP Block Transfer

- PROBLEM: avoid segmentation in the lower layers (IPv6)
- SOLUTION: COAP Block Transfer Mode
  - brings up fragmentation at the application layer

Block transfer



CoAP Client
CoAP Server
CON GET /light
ACK block2(nr=0, m=1, sz=1024) 2.05 "</light>..."
CON block2(nr=1, m=0, sz=1024) GET /light
ACK block2(nr=1, m=1, sz=1024) 2.05 "</light>..."
CON block2(nr=2, m=0, sz=1024) GET /light
ACK block2(nr=2, m=1, sz=1024) 2.05 "</light>..."
CON block2(nr=3, m=0, sz=1024) GET /light
ACK block2(nr=3, m=0, sz=1024) 2.05 "</light>..."
/light (4096 B)

- *Block2* Option added to messages
  - nr=incremental block number within original data
  - m=more blocks flag
  - sz=block size

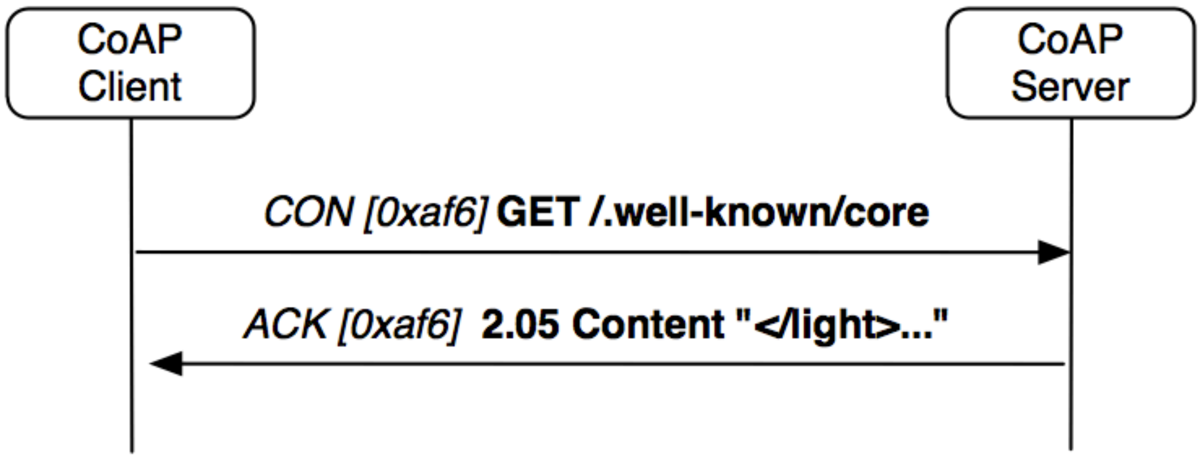
Discovery & Semantics

- Resource Discovery
  - GOAL: 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


CoAP Client
CoAP Server
CON [0xaf6] GET /.well-known/core
ACK [0xaf6] 2.05 Content "</light>..."

```
</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"
```

Getting Started with CoAP

- Open source implementations:
  - Java CoAP Library Californium
  - C CoAP Library Erbium
  - libCoAP C Library
  - jCoAP Java Library
  - OpenCoAP C Library
  - TinyOS and Contiki include CoAP support
- Firefox has a CoAP plugin called Copper
- Wireshark has CoAP plugin