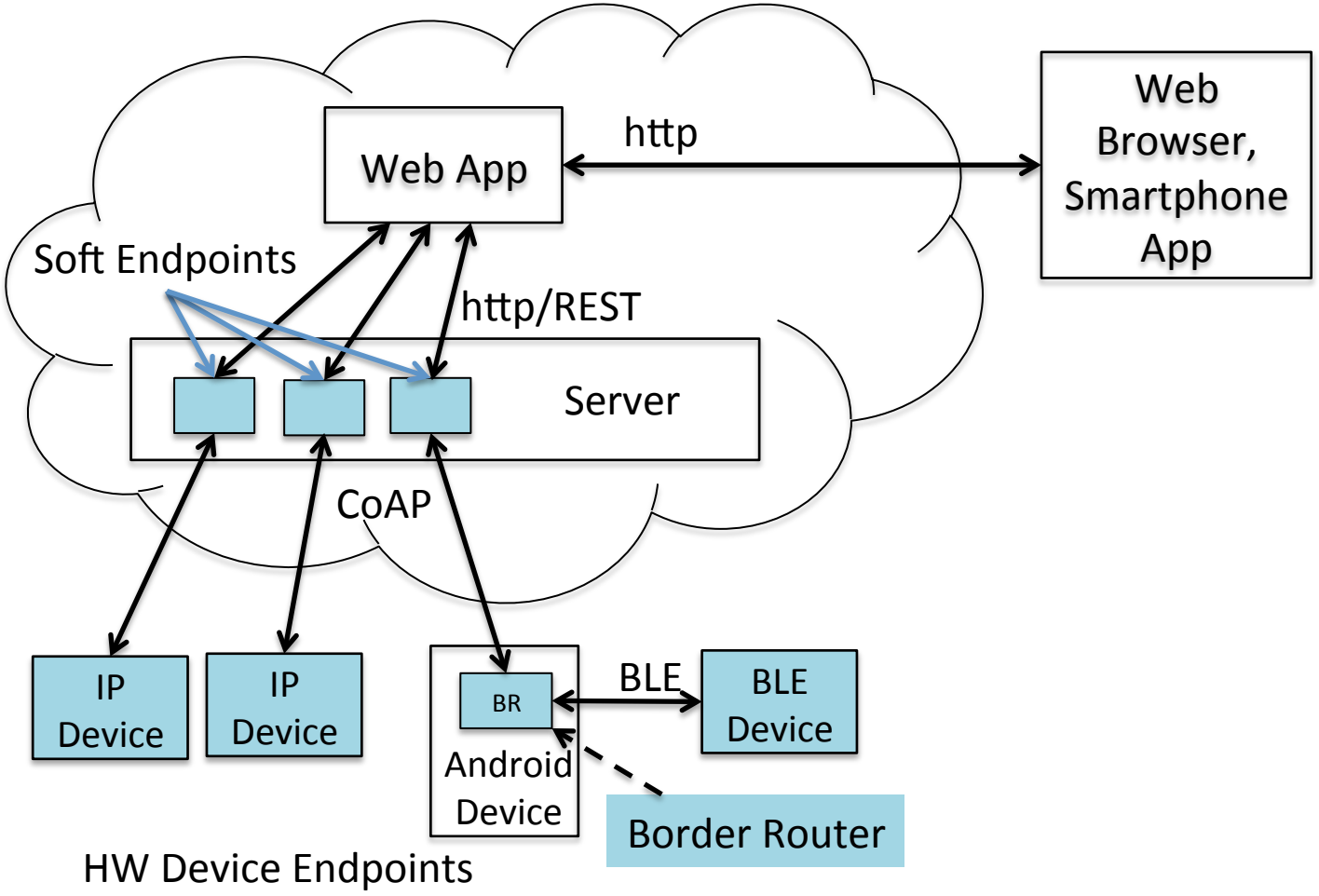
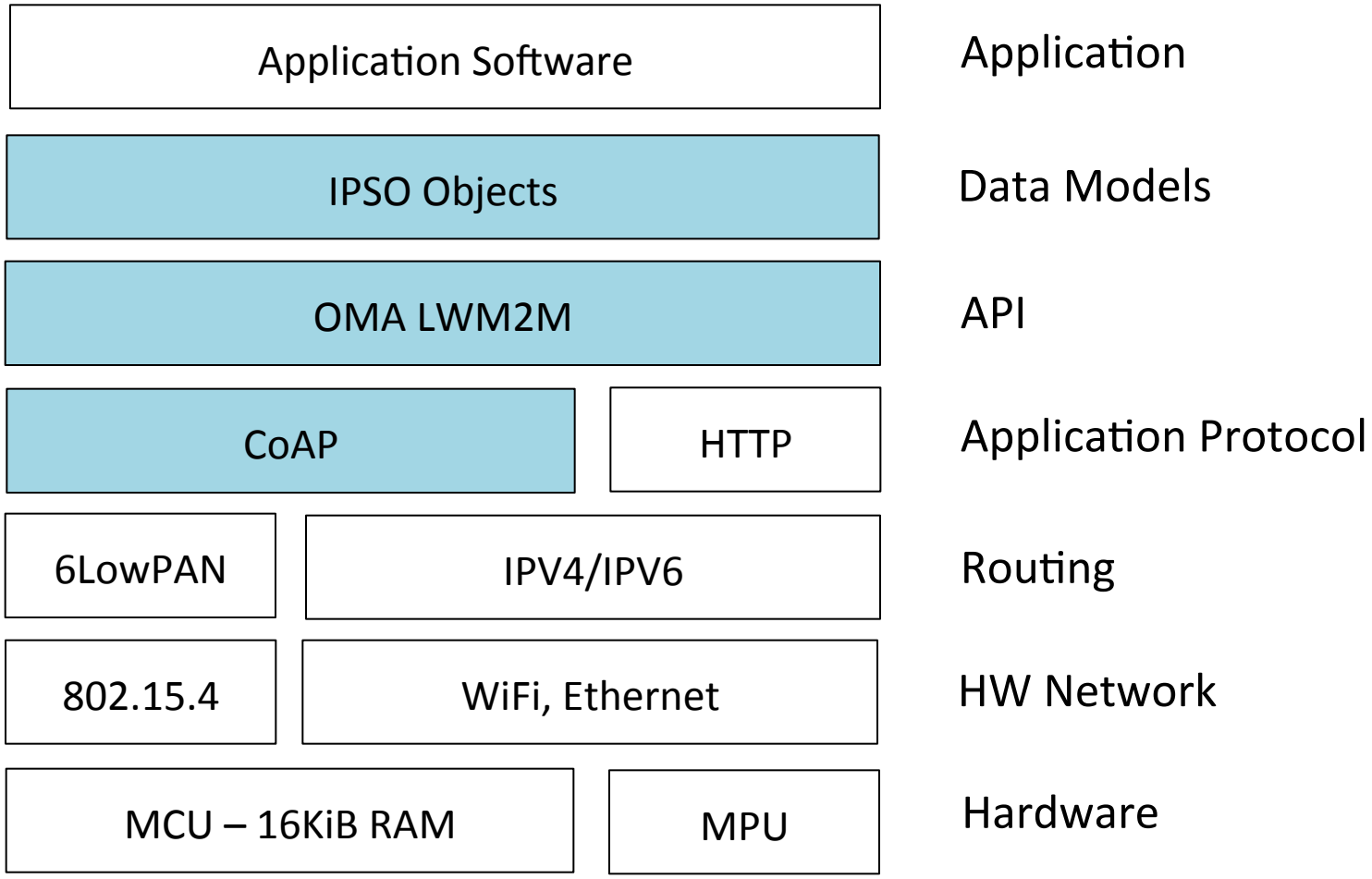
Reference Architecture



Protocol Layers and IoT Standards

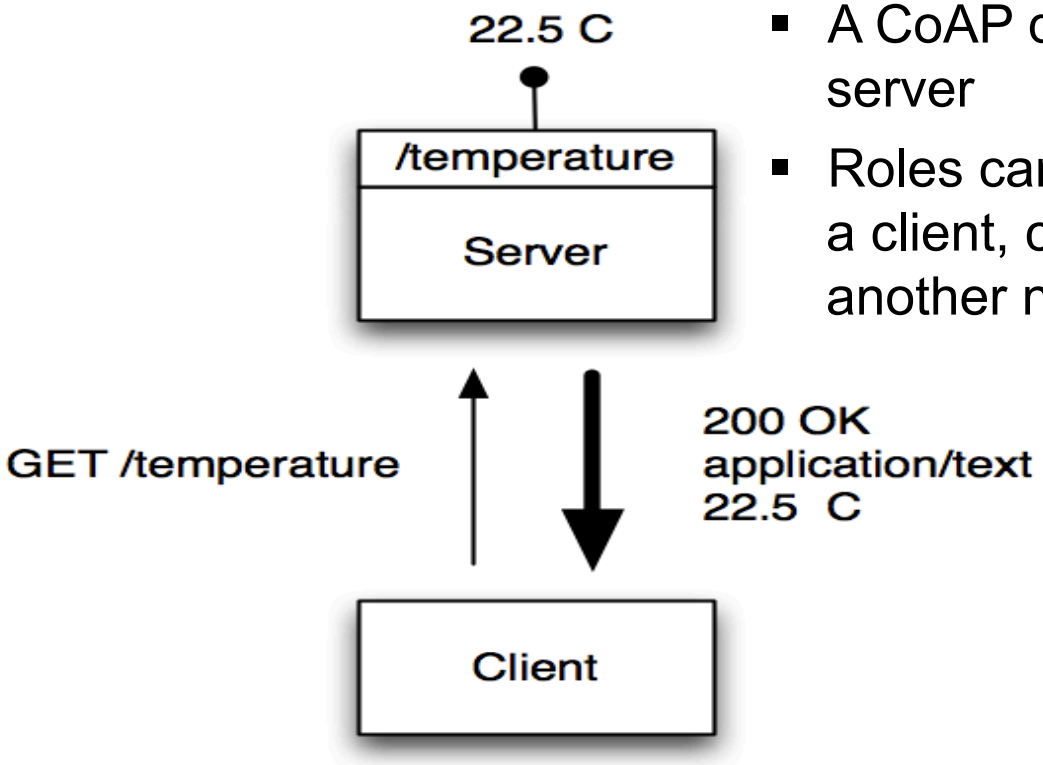


Protocol Layers Work Together

- IPSO Smart Objects are based on OMA LWM2M
  - Defines application objects composed of resources using the LWM2M Object Model
  - Complex objects can be composed from simple objects
  - Easy to add new resource and object types as needed
- OMA LWM2M is based on CoAP
  - Provides a server profile for IoT middleware
  - Defines a simple reusable object model
  - Defines management objects and reuses REST API for onboarding and device life cycle management
- CoAP and related standards from IETF
  - CoAP Protocol (RFC 7252) provides a REST API for device abstraction and data compatibility layer for constrained networks and devices
  - HTTP Proxy provides application abstraction through standard web APIs
  - Core-link-format (RFC 6690) provides a way to add semantic descriptors in the form of web links and enables local resource discovery through the REST API
  - Resource Directory provides an API for scalable discovery and linking using core-link-format primitives

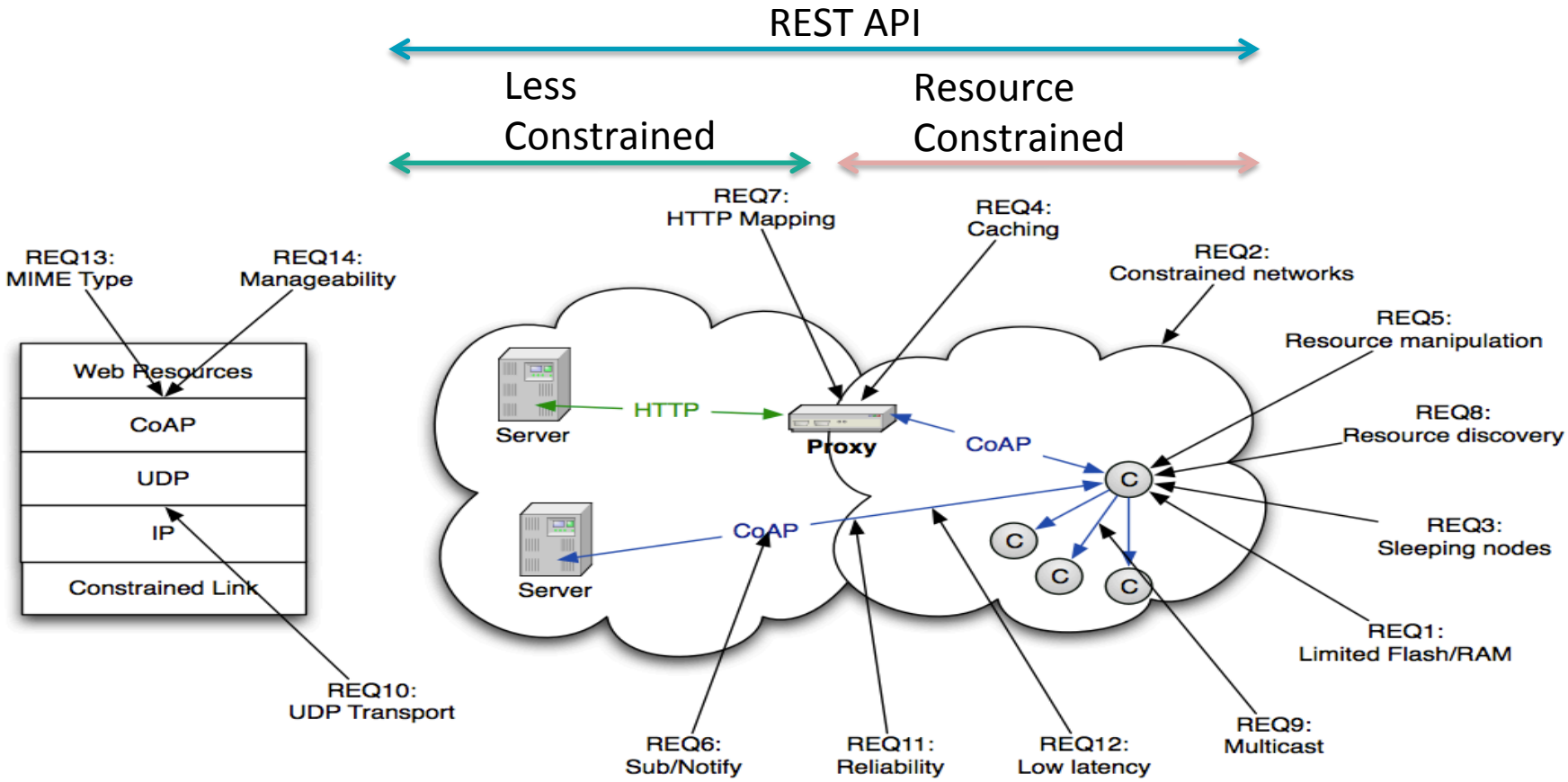
CoAP Protocol

CoAP is REST for Constrained Devices



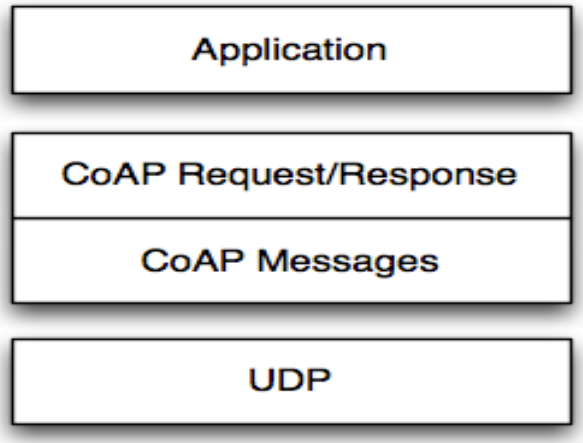
- Makes each device a lightweight server that exposes a REST API
- A CoAP device can be both client and server
- Roles can be reversed and the sensor, as a client, can update a REST API at another node, device or server

CoAP Use Case Requirements

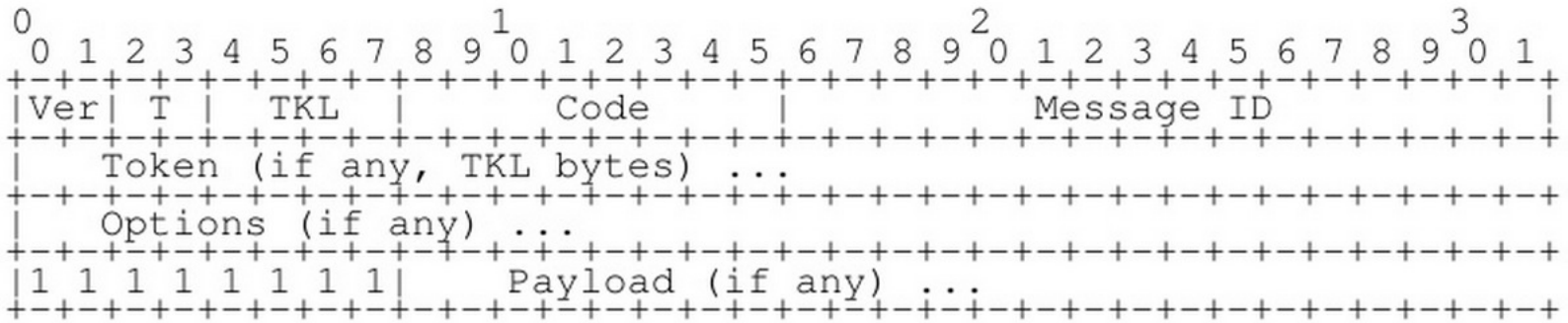


Transport Model

- 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 mapped onto CoAP Messages
  - Method, Response Code and Options (URI, content-type etc.) define REST exchanges, very similar to HTTP (HTTP 404 response semantics (not found) mapped to CoAP 4.04 response code)
- Asynchronous Notifications
  - Observer option for GET allows asynchronous state update responses from a single request



CoAP Header – Binary Protocol Mapping



Ver - Version (1)

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

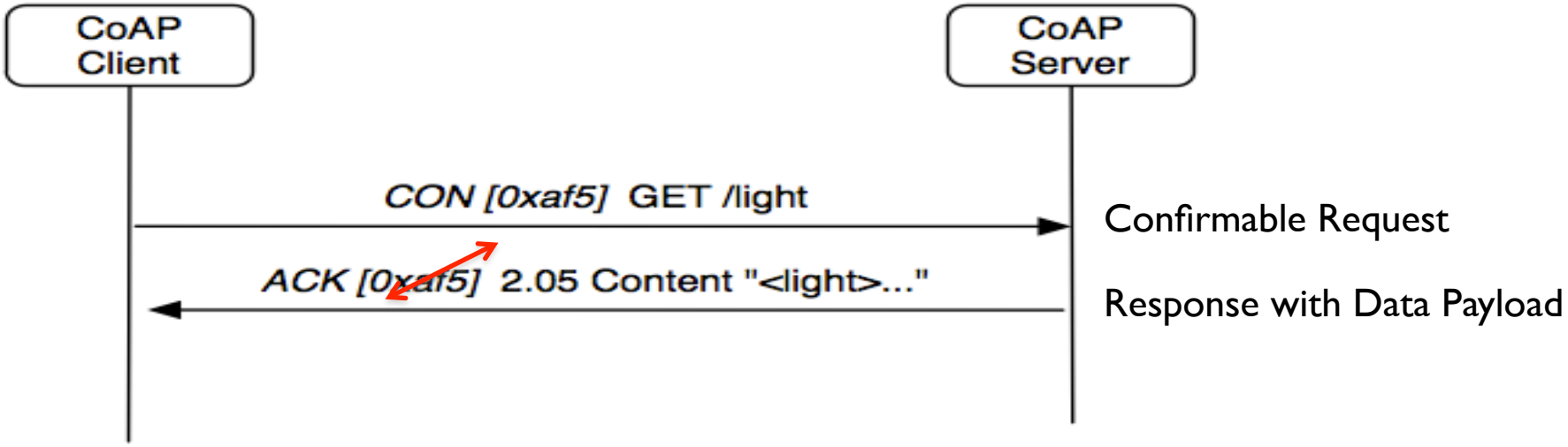
TKL- Token Length, if any, the number of Token bytes after this header

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

Message ID - 16-bit identifier for matching responses

Token - Optional response matching token

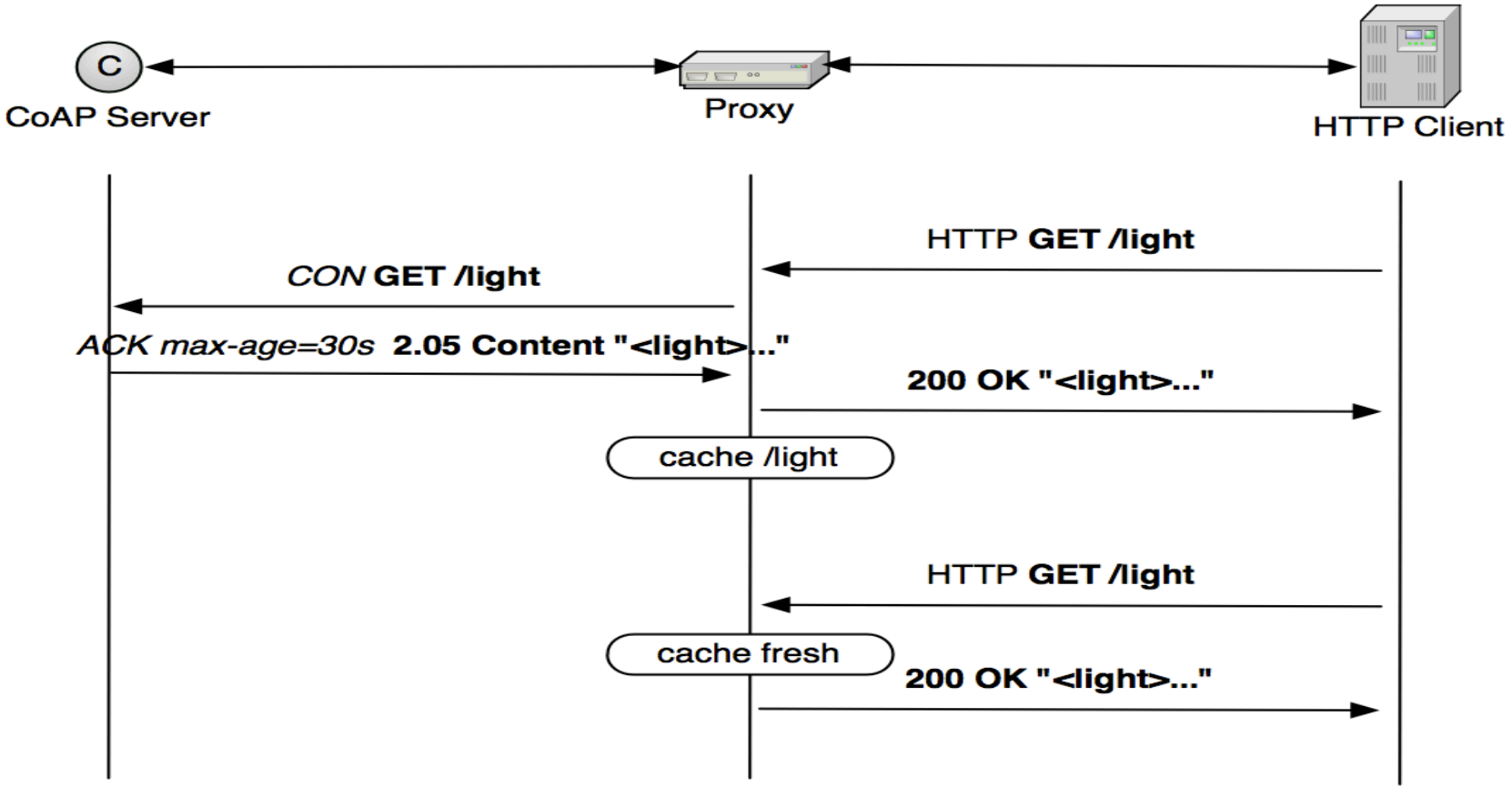
CoAP Request - Response



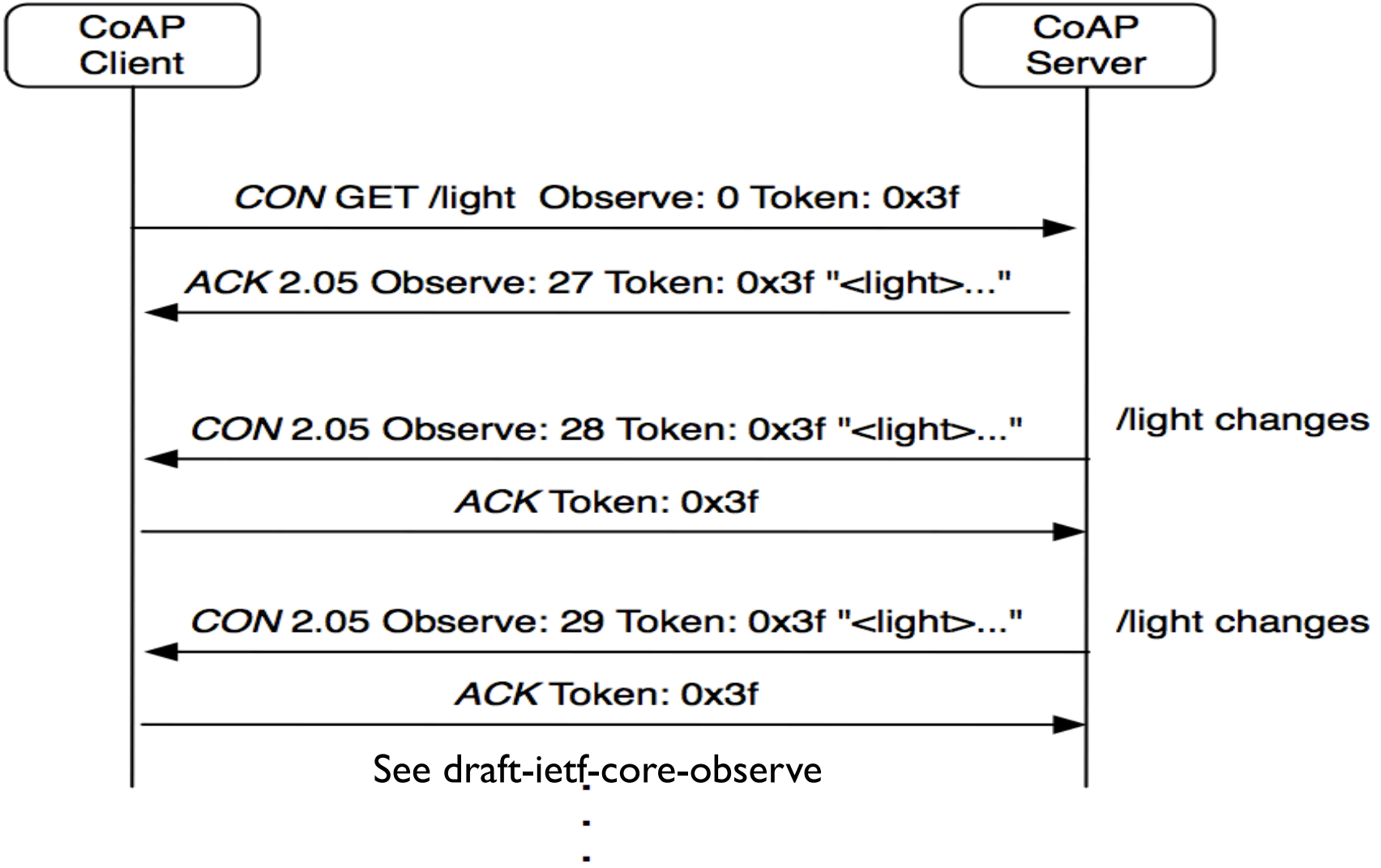
CoAP Caching

- CoAP includes a simple caching model
  - Cacheability determined by response code
  - An option number mask determines if it is a cache key
  - Resource Discovery returns links to cache/proxy
- 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, a node behind a firewall,
  - or to reduce network load and battery drain

CoAP Proxy Caching



CoAP Observe – Asynchronous Notification



CoAP Semantic Links & Discovery

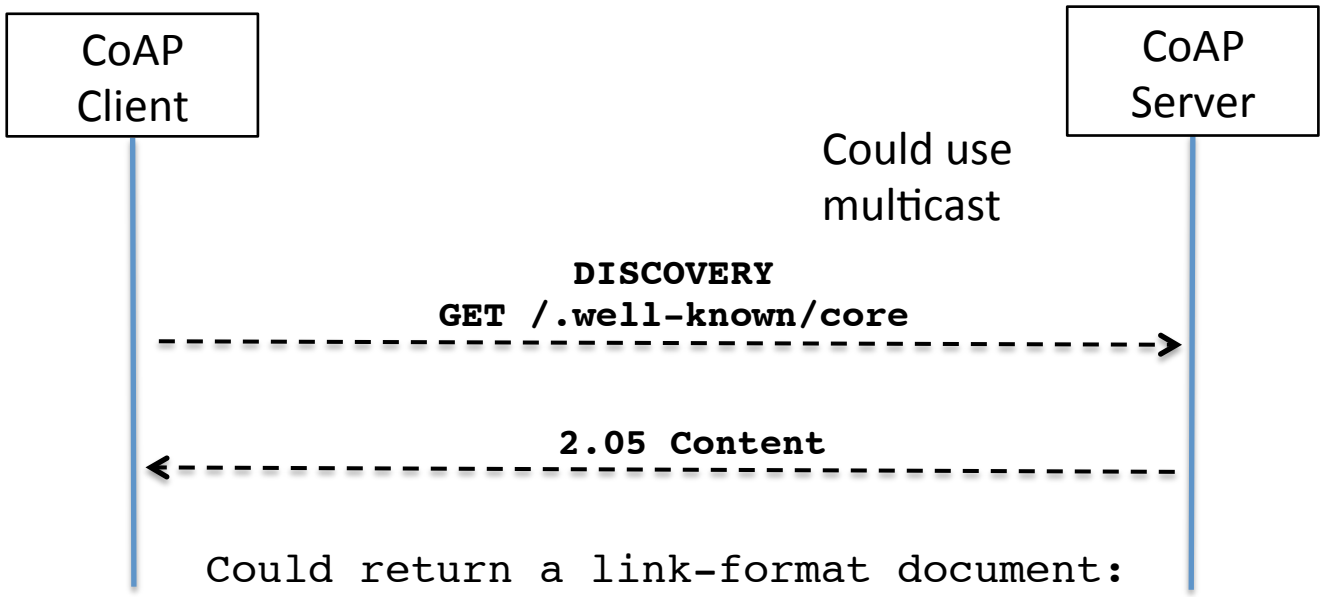
Discovery

- Service Discovery
  - What services are available in the first place?
  - Goal of finding the IP address, port and protocol
  - Can be performed by e.g. DNS-SD when DNS is available
  - CoAP Services can be discovered using Resource Discovery
- Resource Discovery
  - What are the Web resources I am interested in?
  - Goal of finding URLs to link into applications
  - Performed using Web Linking and REST API
    - CoRE Link Format is designed to enable resource discovery

Web Linking for Machines

- RFC6690 is aimed at Resource Discovery for M2M
  - Defines semantic link serialization and content-types suitable for M2M
  - Defines a well-known resource where links are stored
  - Enables query string parameters for discovery by attribute and relation
  - 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-format document
  - URL, resource type, interface type, content-type, size are some basic relations

Local Network Discovery

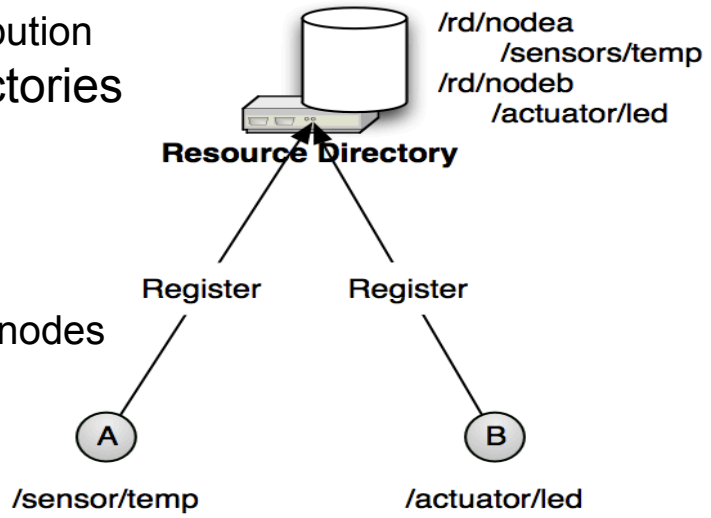



```
</3//9>;obs;rt="urn:X-ipso:batt-level";ct="50",
</3//0>;rt="urn:X-ipso:dev-mdl";ct="50",
</3//1>;rt="urn:X-ipso:dev-mfg";ct="50",
</3305/0/5800>;obs;rt="urn:X-ipso:pwr-w";ct="50",
</3305/0/5805>;obs;rt="urn:X-ipso:pwr-accum-wh";ct="50",
</3303/0/5700>;obs;rt="urn:X-ipso:temp-C";ct="50"
```

Resource Directory Discovery

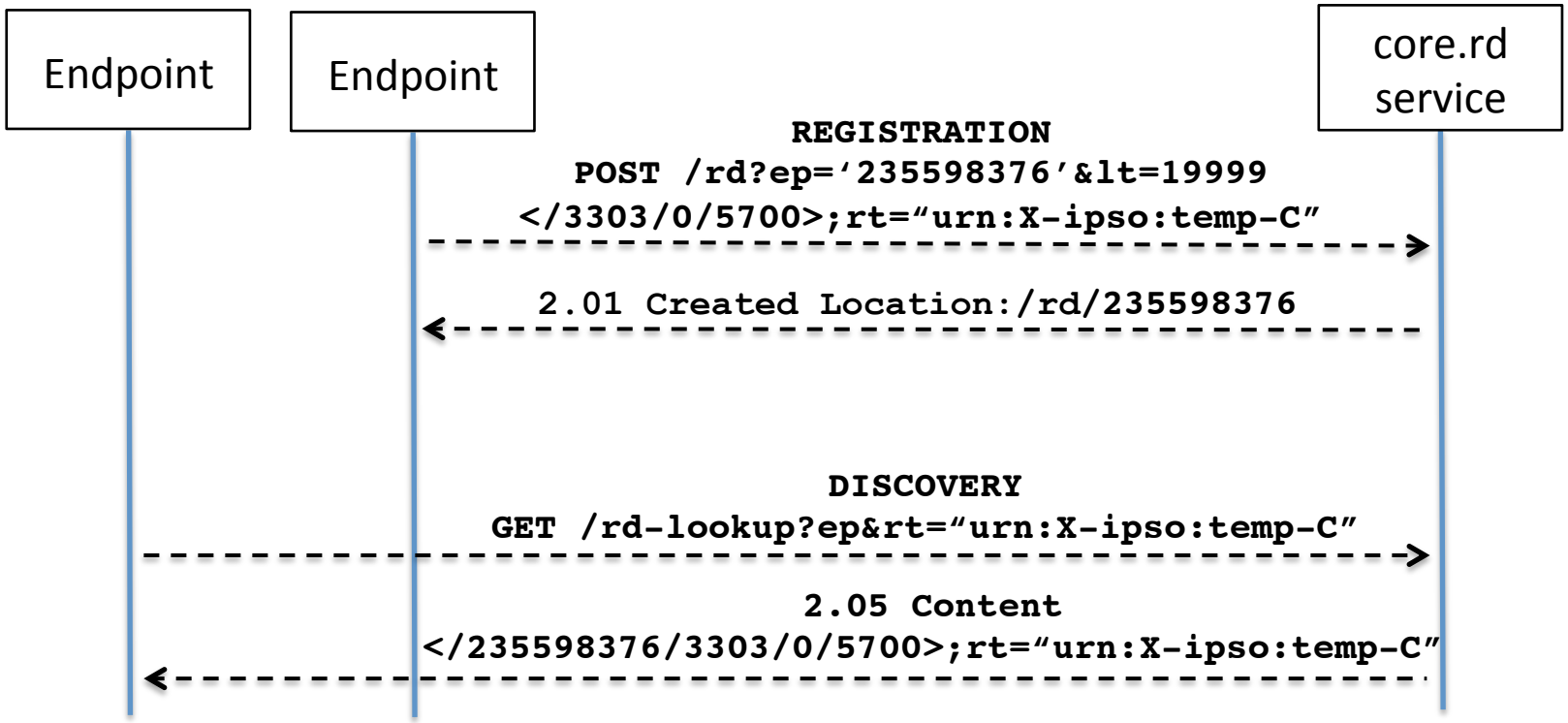
- CoRE Link Format defines
  - The link format
  - Peer-to-peer discovery
- A directory approach is also useful
  - Supports sleeping nodes
  - No multicast traffic, longer battery life
  - Remote lookup, hierarchical and federated distribution
- CoRE Link Format is used in Resource Directories
  - Nodes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

Web Applications Discover Registered Resources



See draft-ietf-core-resource-directory

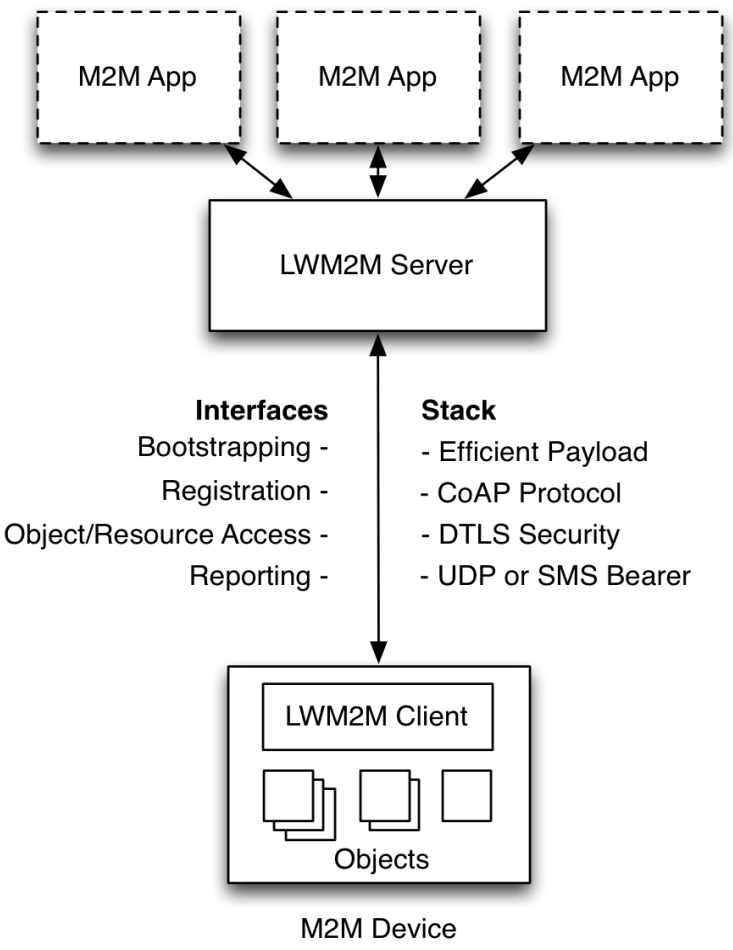
Resource Directory



See draft-ietf-core-resource-directory

OMA LWM2M

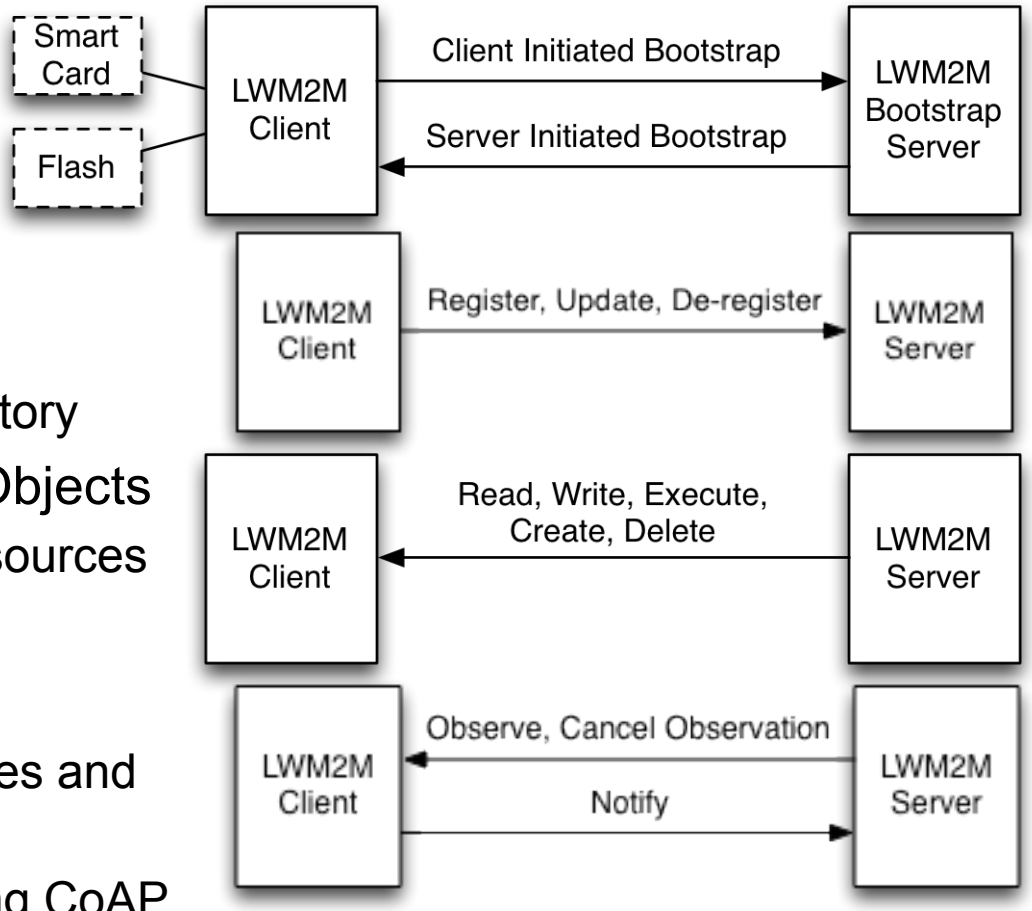
LWM2M Architecture



- Web Applications
  - Application abstraction through REST API
  - Resource Discovery and Linking
- LWM2M Server
  - CoAP
  - HTTP Caching Proxy
  - Resource Directory
  - Gateway and Cloud deployable
- LWM2M Clients are Devices
  - Device abstraction through CoAP
  - LWM2M Clients can be CoAP Servers
  - Any IP network connection

LWM2M Interfaces

- Bootstrap Interface
  - Configure Servers & Keying
  - Pre-Configure, Smart Card, Server Initiated Bootstrap
- Registration Interface
  - RFC6690 and Resource Directory
- Management Interface Using Objects
  - Management Objects and Resources
  - Uses the CoAP REST API
- Reporting Interface
  - Subscription to Object Instances and Resources
  - Asynchronous notification using CoAP

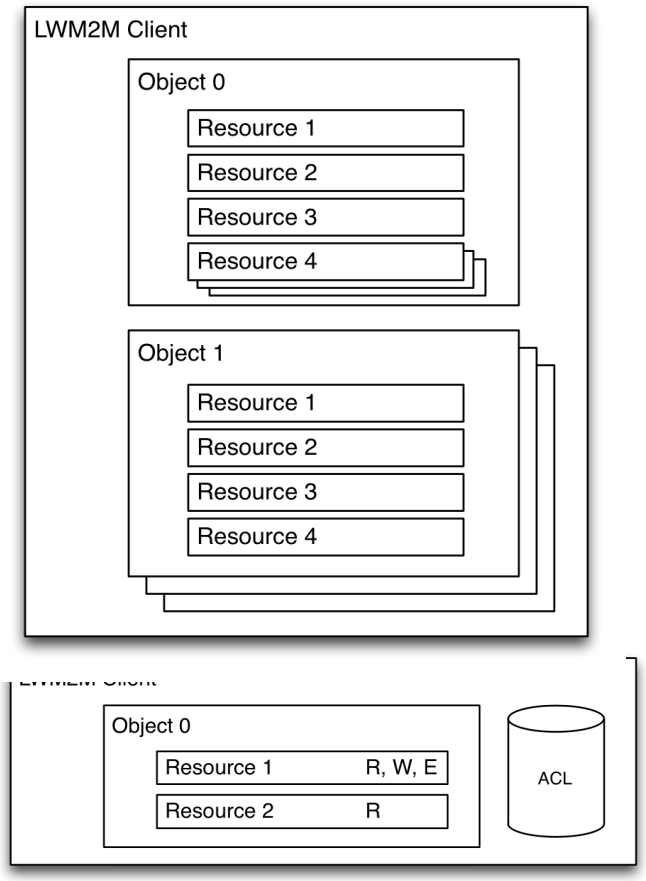


LWM2M Object Model

- A Client has one or more Object Instances
- An Object is a collection of Resources
- A Resource is an atomic piece of information that can be
  - Read, Written or Executed
- Resources can have multiple instances
- Objects and Resources are identified by a 16-b Integer, Instances by an 8-bit Integer
- Objects/Resources are accessed with simple URIs:

  /{Object ID}/{Object Instance}/{Resource ID}

  Ex: /3/0/1 (Object Type=Device, Instance=0, Resource=Device Mfg.)



LWM2M Management Objects

Object	Object ID
LWM2M Security	0
LWM2M Server	1
Access Control	2
Device	3
Connectivity Monitoring	4
Firmware	5
Location	6
Connectivity Statistics	7

Example Object – Position Object

Resource Name	ID	Access Type	Multiple Instances?	Type	Range	Units	Descriptions
Latitude	0	R	No	Decimal		Deg	The decimal notation of latitude, e.g. -43.5723 [World Geodetic System 1984]
Longitude	1	R	No	Decimal		Deg	The decimal notation of longitude, e.g. 153.21760 [World Geodetic System 1984]
Altitude	2	R	No	Decimal		m	The decimal notation of altitude in meters above sea level.
Uncertainty	3	R	No	Decimal		m	The accuracy of the position in meters.
Velocity	4	R	No	Refers to 3GPP GAD specs		Refers to 3GPP GAD specs	The velocity of the device as defined in 3GPP 23.032 GAD specification. This set of values may not be available if the device is static.
Timestamp	5	R	No	Time			The timestamp of when the location measurement was performed.

IPSO Smart Objects

IPSO Alliance

- IPSO Alliance - semiconductor, device, and system manufacturers
- Smart Object Committee meets every 2 weeks, recently completed the Smart Objects 1.0 IPSO Technical Guideline
- Uses the OMA LWM2M Object Model to define application objects
- Resource IDs and Object IDs are registered with the OMNA
- Provides a framework for Application Level Interoperability
- Builds common definitions of web objects for use with standard web protocols (CoAP, HTTP)
- Defines reusable resources and objects that map to physical sensors, actuators, objects

IPSO Smart Objects

- IPSO Smart Objects provide a common data model across different sensor types, enabling application level interoperability
- Interoperability between end devices and applications
- Allows decoupling of devices from dedicated application services
- Repurposing and multi-purposing of devices, reusability of application software
- Interoperability across platforms and M2M protocols
- Enables developers of embedded device and web services to focus on the value endpoints

IPSO Smart Object

- Common representations and units
- Refers to well-known namespaces like ucum
- Default units e.g. Celsius, kPa
- Object, object instances, and resource instances are addressable using paths constructed from object and resource type IDs
- Example: GET /sensors/3303/0/5700
- Returns a representation of the current value of the ‘5700’ resource (Current Value) from instance ‘0’ of Object Type ‘3303’ (Temperature)

Example Smart Object - Temperature

Object info

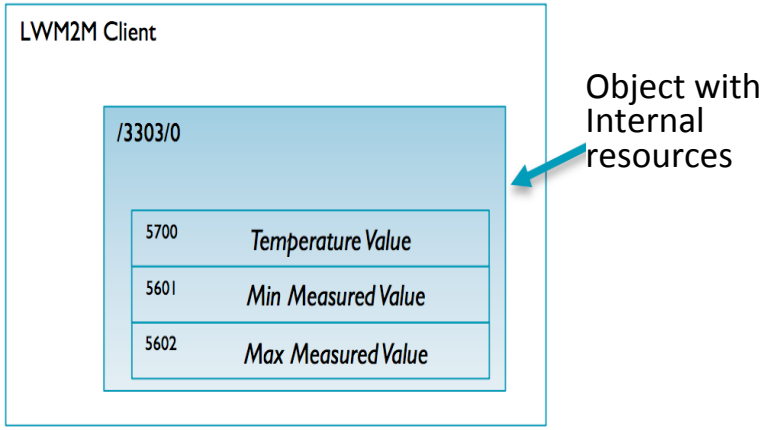
Object	Object ID	Object URN	Multiple Instances?
IPSO Temperature	3303	urn:oma:lwm2m:ext:3303	Yes

Resource Info

Resource Name	Resource ID	Access Type	Multiple Instances?	Type	Units	Descriptions
Sensor Value	5700	R	No	Decimal	Cel	This resource type returns the Temperature Value in °C
Min Measured Value	5601	R	No	Decimal	Cel	The minimum value measured by the sensor since it is ON
Max Measured Value	5602	R	No	Decimal	Cel	The maximum value measured by the sensor since it is ON

Accessing the Resources

- *Temperature Value* */3303/0/5700*
- *Min Measured Value* */3303/0/5601*
- *Max Measured Value* */3303/0/5602*



Object Annotation

- Semantic annotation (not part of LWM2M Standard) uses core-link-format metadata for semantic description
- Described in IETF CoRE Interfaces document (http://datatracker.ietf.org/doc/draft-ietf-core-interfaces) enables GET by relation and attribute
  - For example, GET /rd-lookup?ep&rt="urn:X-ipso:temp-C"

```
Returns: </sensors/3303/0/5700>;obs;if="urn:X-ipso:sensor";rt="urn:X-ipso:temp-C";ct=50;u="ucum:degC"
```

- Refers to qualified, resolvable namespaces and concepts
- Supports CoRE Resource Discovery

Example Smart Obejct Ad-Hoc binding

Object info:

Object	Object ID	Object URN	Multiple Instances?	Description
Heart Rate	12200	urn:oma:lwm2m:x:12200	Yes	Heart Rate Monitor

Resource Info:

Resource Name	Resource ID	Access Type	Multiple Instances?	Mandatory	Type	Range or Enumeration	Units	Descriptions
Sensor Value	5700	R	No	Mandatory	Float		BPM	Heart Rate Measurement Value
Digital Input State	5500	R	No	Optional	Boolean			Sensor contact status 0=no contact, 1= contact
Total Energy	5950	R	No	Optional	Float		kJ	Energy Expended
Reset Cumulative Energy	5822	E	No	Optional	Opaque			Reset 5950 Energy Expended to zero
Body Sensor Location	5951	R,W	No	Optional	String			Intended sensing location on the body
R-R Interval	5952	R	No	Optional	String			Sequence of R-wave intervals

Binding to Smart Thermostat

Object info:

Object	Object ID	Object URN	Multiple Instances?	Description
Smart Thermostat	12300	urn:oma:lwm2m:x:12300	Yes	Smart Thermostat with multiple settings

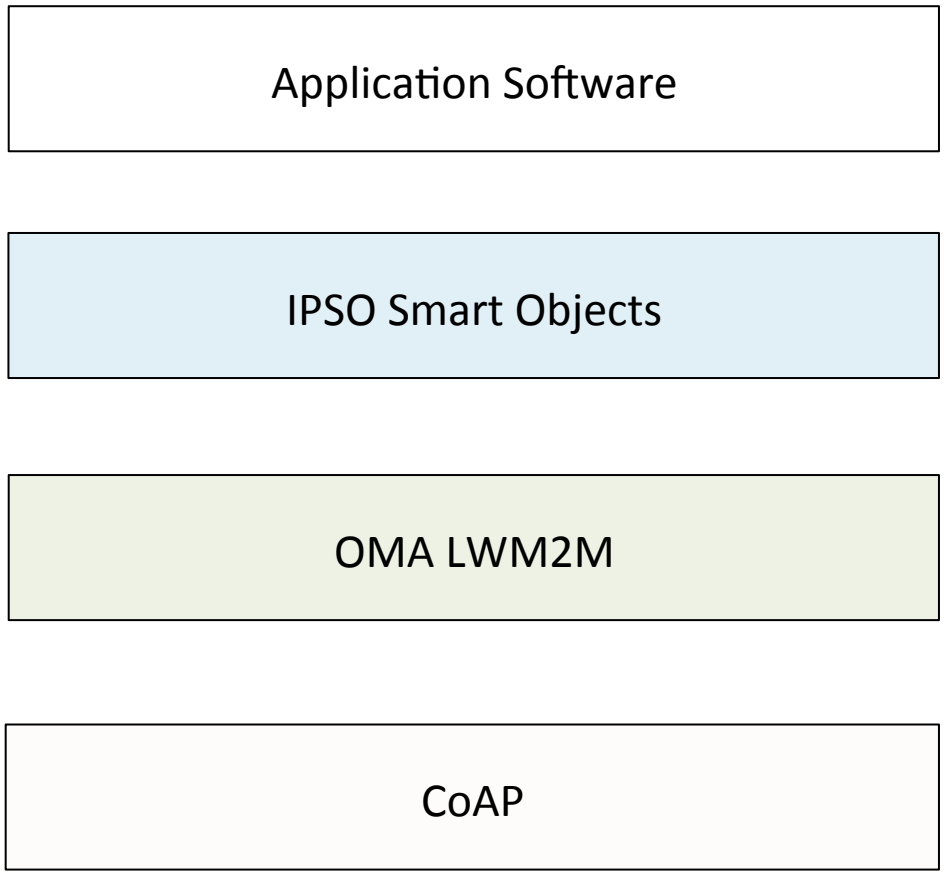
Resource Info:

Resource Name	Resource ID	Access Type	Multiple Instances ?	Mandatory	Type	Range or Enumeration	Units	Descriptions
Sensor Value	5700	R	No	Mandatory	Float		Per Units resource	Temperature measurement
Units	5500	R,W	No	Mandatory	String	ucum:degF, ucum:degC		Units for 5700
Application Type	5750	R,W	No	Optional	String			Name, e.g. "Hall Thermostat"
Cooling	5200	R	No	Optional	Boolean			1=cooling
Heating	5201	R	No	Optional	Boolean			1=heating
Heat Source	5203	R	No	Optional	String	"Emergency", "Normal"		Indicates heat source
Fan Timer Active	5204	R,W	No	Optional	Boolean			1=running
Fan Timeout	5205	R,W	No	Optional	String		UTS	Time for fan to stop
Energy Save Mode	5206	R,W	No	Optional	Boolean			1= Energy Save mode
Away Mode	5207	R,W	No	Optional	Boolean			0=Home, 1=Away
Setpoint	5208	R	No	Optional	Float			Desired Temperature
HVAC Mode	5209	R,W	No	Optional	String	"Heat", "Cool", "Heat-Cool"		System Mode
High Setpoint	5210	R,W	No	Optional	Float			Highest desired temperature
Low Setpoint	5211	R,W	No	Optional	Float			Lowest desired temperature
High Away Setpoint	5212	R,W	No	Optional	Float			Highest away mode temperature
Low Away Setpoint	5213	R,W	No	Optional	Float			Lowest away mode temperature

Smart Object Summary

- Simple web objects that represent common sensors, actuators, and data elements exposed for Internet of Things applications
- Based on Internet and Industry Standards
- Objects can be composed into more complex objects; for example a temperature object, set point object, and load control object can be combined to create a thermostat object
- About to publish the first set of objects to cover some common Smart Home and Sensor use cases
- New objects are easy to create; we are planning a developer-friendly process for creating and registering new objects

Layered Standards



Not tied to specific device or protocol
Any Programming Language
Runs on devices, gateways, and services

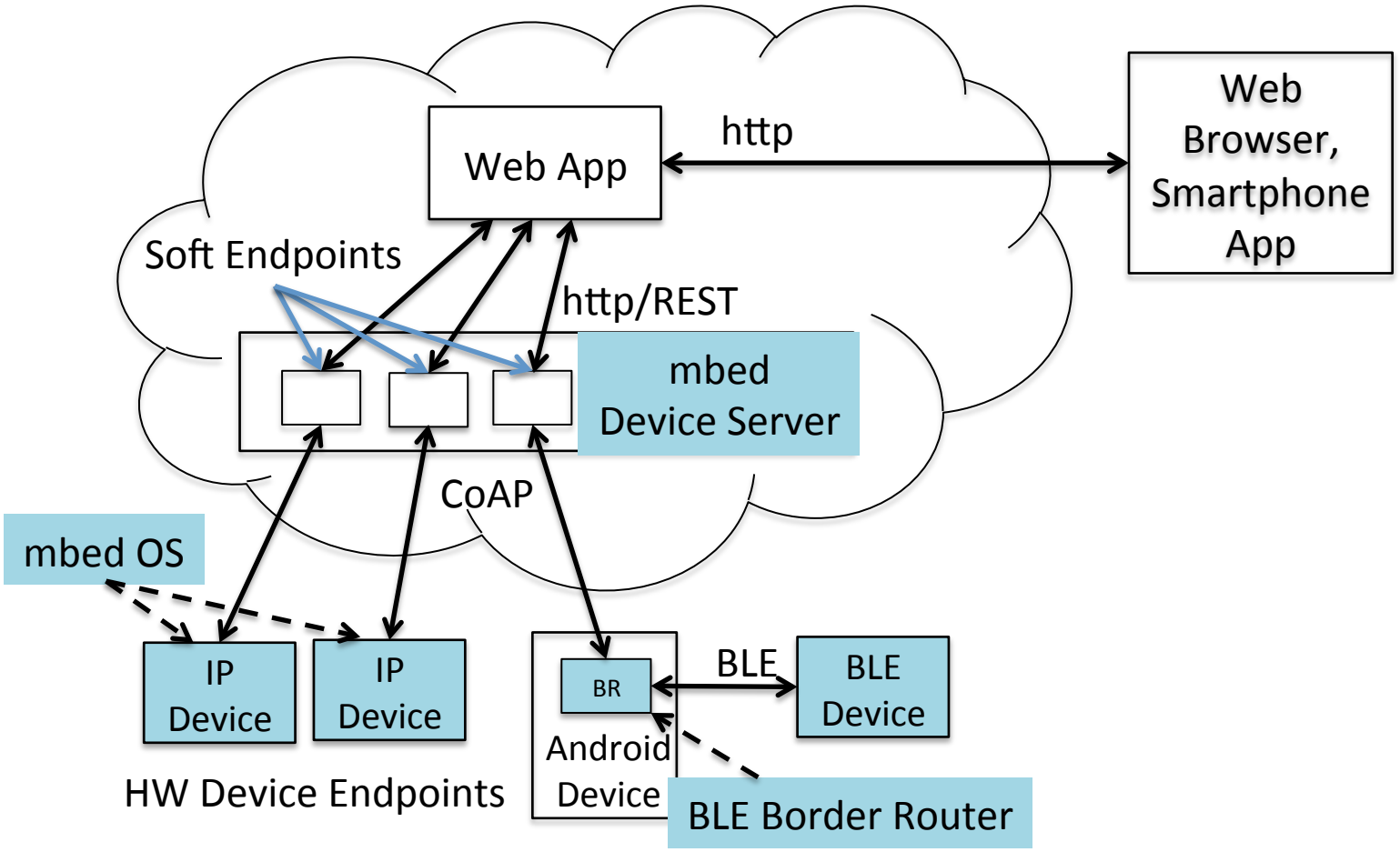
Application Level Interoperability
Reusable Device to Application API
Not tied to any specific protocol

Service Layer Specification
Device Management over CoAP
Object Model for DM and Applications

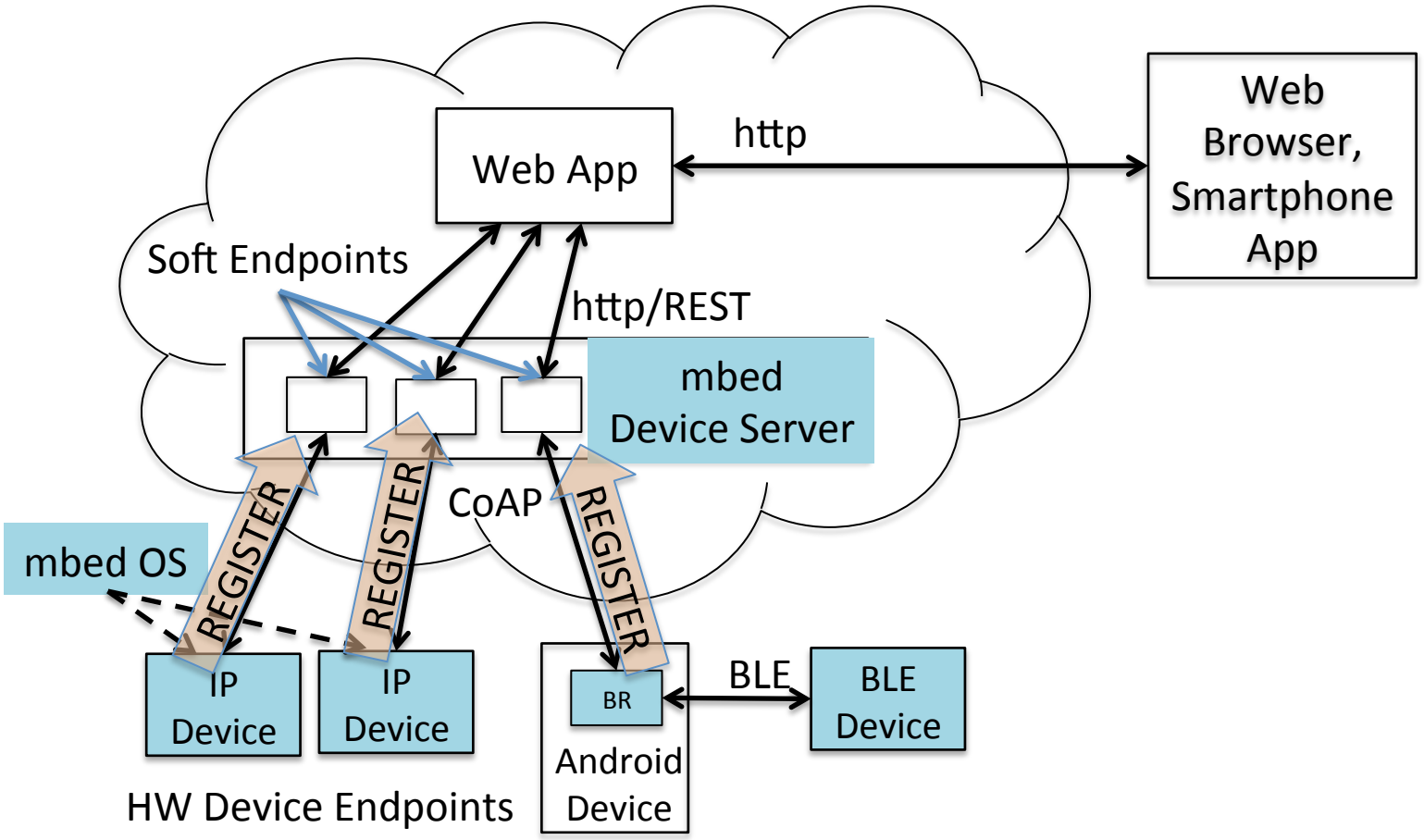
REST protocol for constrained devices
IETF Standard (RFC 7252)
Uses TCP or UDP, any IP connection
Discovery using IP Multicast or Directory

OMA LWM2M and ARM mbed

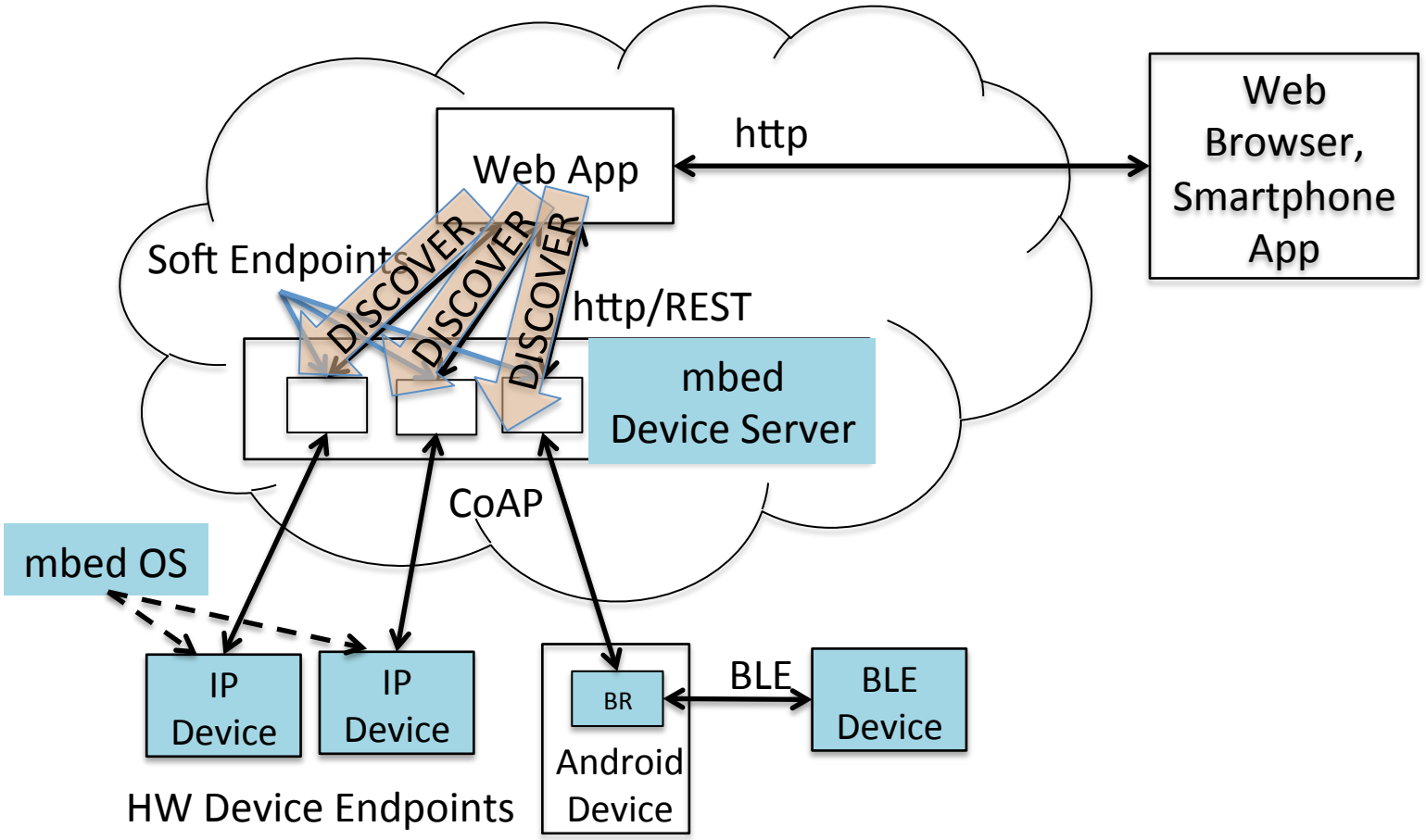
mbed Mapping to Reference Architecture



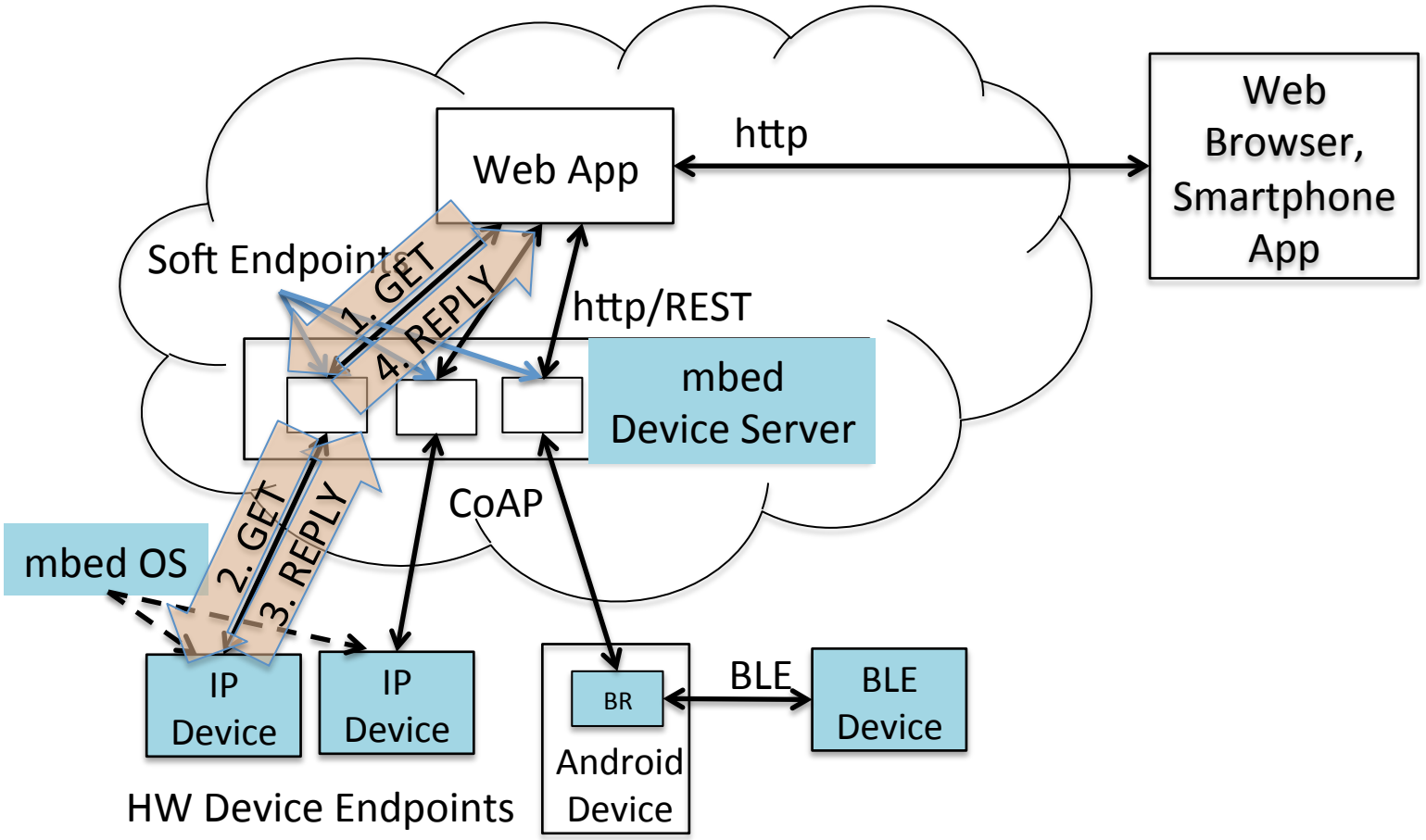
Registration



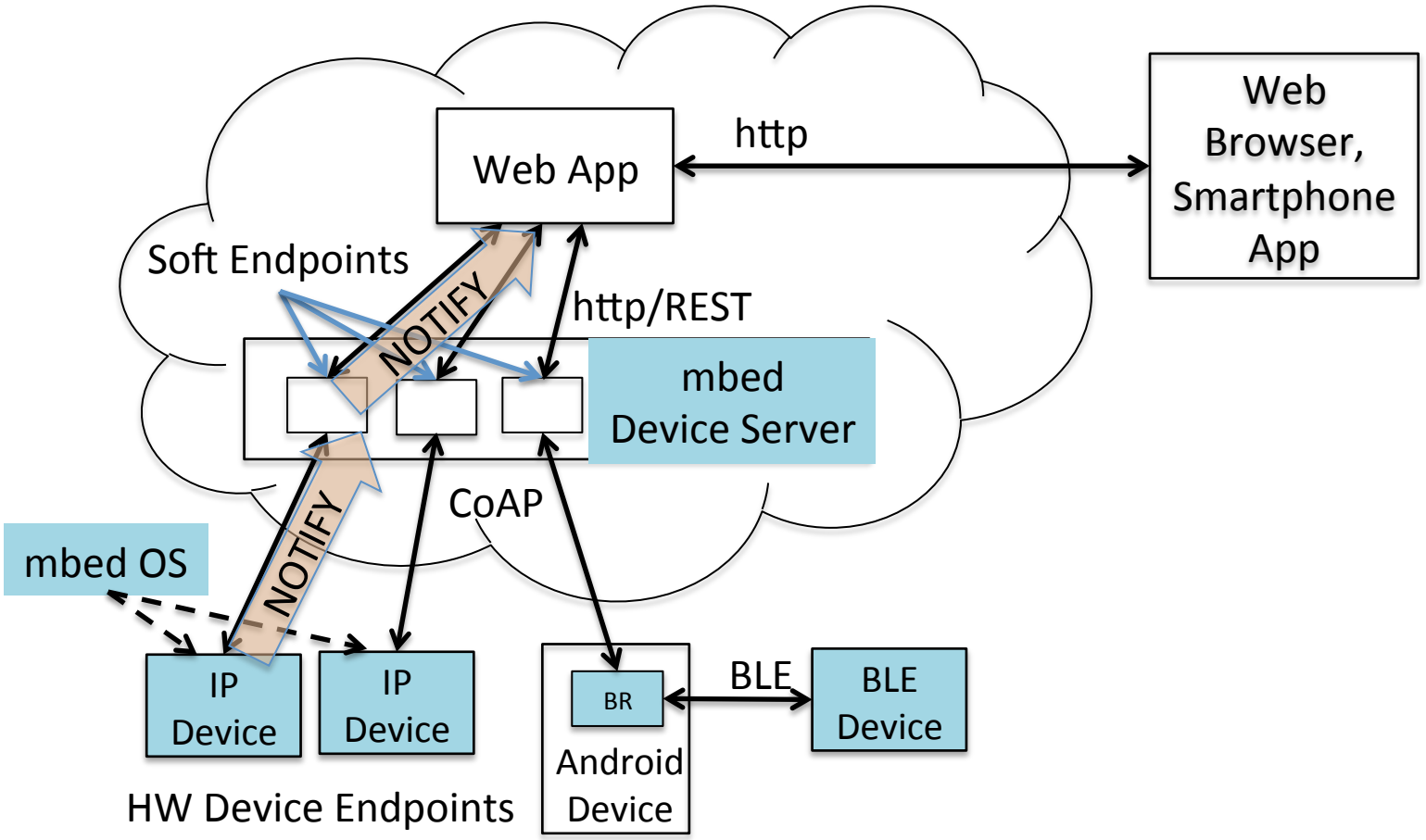
Discovery


Web App
http
Web Browser, Smartphone App
Soft Endpoints
DISCOVER
DISCOVER
DISCOVER
http/REST
mbed Device Server
CoAP
mbed OS
IP Device
IP Device
BR
BLE
BLE Device
Android Device
HW Device Endpoints

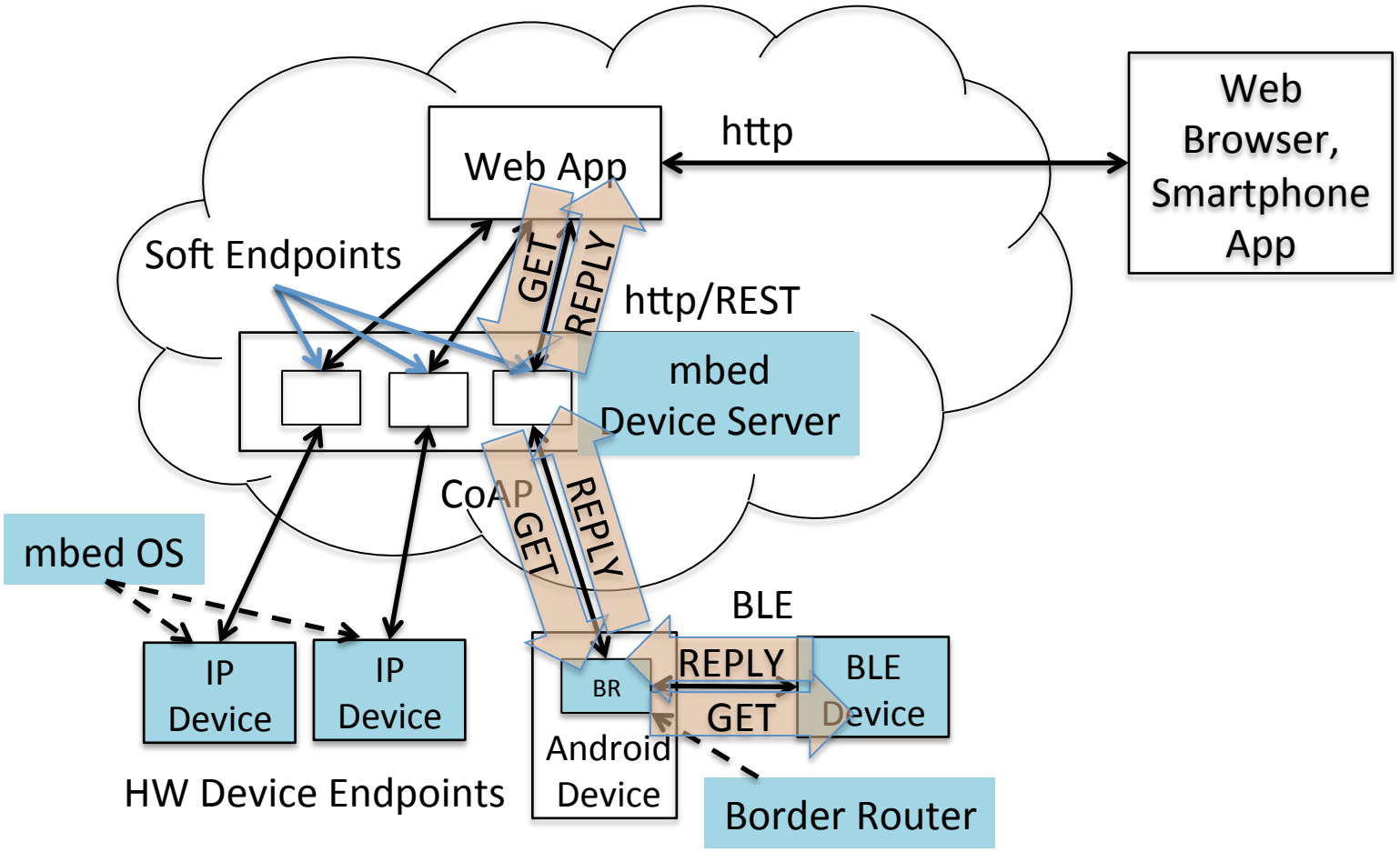
Example Transaction



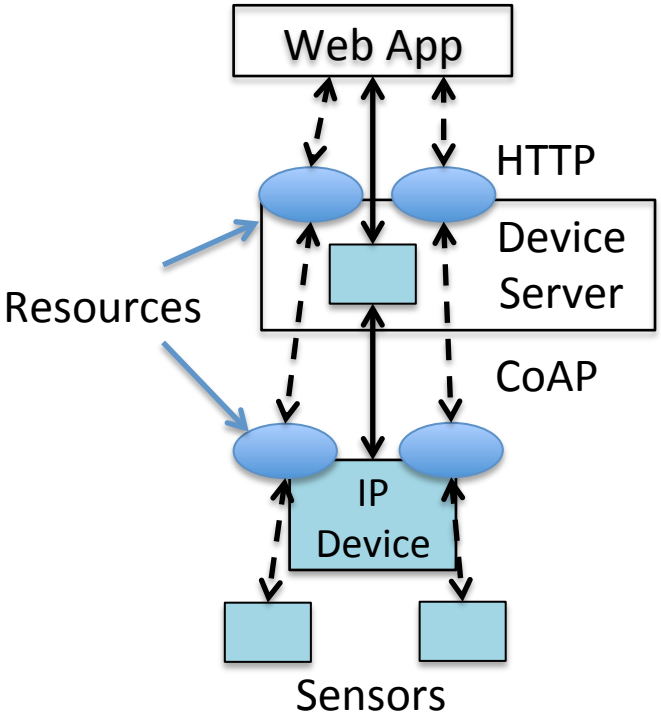
Asynchronous Notification



CoAP over BLE



Resources and Data Models



Resource:

- An addressable element of a REST API
- Has a path or URI
- May represent a sensor or actuator
- Has a set of operations e.g. GET, PUT
- Exposed by devices, registered with server
- Consumed by applications through server

The Data Model defines:

- How the path is constructed - URI template
- Which operations are allowed
- Other properties like data type

Smart Object Data Model - Temperature

Object info

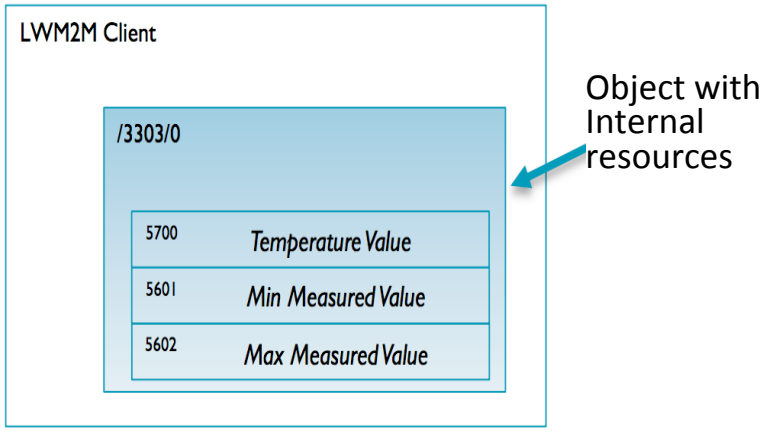
Object	Object ID	Object URN	Multiple Instances?
IPSO Temperature	3303	urn:oma:lwm2m:ext:3303	Yes

Resource Info

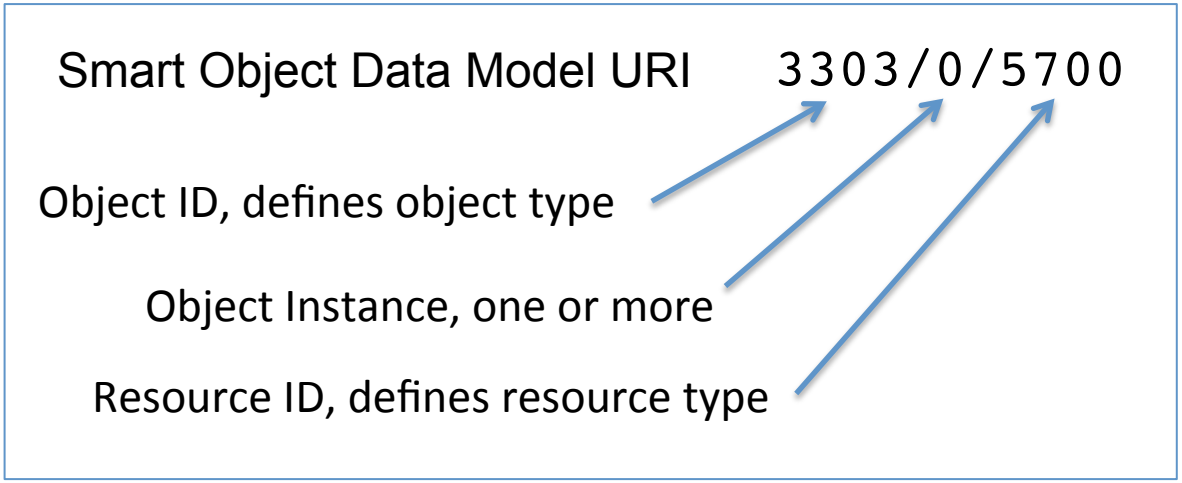
Resource Name	Resource ID	Access Type	Multiple Instances?	Type	Units	Descriptions
Sensor Value	5700	R	No	Decimal	Cel	This resource type returns the Temperature Value in °C
Min Measured Value	5601	R	No	Decimal	Cel	The minimum value measured by the sensor since it is ON
Max Measured Value	5602	R	No	Decimal	Cel	The maximum value measured by the sensor since it is ON

Accessing the Resources

- *Temperature Value* */3303/0/5700*
- *Min Measured Value* */3303/0/5601*
- *Max Measured Value* */3303/0/5602*



Smart Object Resource Design



- Objects represent single points of interest, for example a temperature senor
- An object may have multiple instances
- Resources represent attributes of the object, for example the last measured value, the smallest measured value, the greatest measured value
- Object and resource IDs are meant to be reusable, representing common measurements and concepts
- Please see the IPSO Smart Object Guideline

Smart Object Starter Pack

Table 1 Smart Objects defined by this Technical Guideline

Object	Object ID	Multiple Instances?
IPSO Digital Input	3200	Yes
IPSO Digital Output	3201	Yes
IPSO Analogue Input	3202	Yes
IPSO Analogue Output	3203	Yes
IPSO Generic Sensor	3300	Yes
IPSO Illuminance Sensor	3301	Yes
IPSO Presence Sensor	3302	Yes
IPSO Temperature Sensor	3303	Yes
IPSO Humidity Sensor	3304	Yes
IPSO Power Measurement	3305	Yes
IPSO Actuation	3306	Yes
IPSO Set Point	3308	Yes
IPSO Load Control	3310	Yes
IPSO Light Control	3311	Yes
IPSO Power Control	3312	Yes
IPSO Accelerometer	3313	Yes
IPSO Magnetometer	3314	Yes
IPSO Barometer	3315	Yes

Device Programming for Resources

- The mbed library for LWM2M and Smart Objects provides resource classes (e.g. DynamicResource)
- **Two files** are involved in resource programming
- Resources are implemented in a **<resource>.h** file for each addressable resource, which is a wrapper for the generic class constructor and resource-specific code
- Resources are configured and created in the main program (e.g. **main.cpp**)
- Resources are registered with the Device Server when the endpoint start() is called, usually in main.cpp

Device Programming for Resources

- Refer to the lab note on resource programming for specific instructions
- Some attributes and parameters need to be customized for each device
  - Endpoint Name
  - Registration Domain
  - Device Server IP address
- And for each resource
  - Resource.h file
  - Resource path
  - Max-age to control cache lifetime - optional
  - Observation sample time - optional

Device Server API

- Devices register resources with a Device Server by uploading links
- Registrations have a lifetime and are refreshed periodically by the device
- The Device Server is a proxy for resources that are registered by devices
- Applications can access device resources by using the HTTP proxy, which exposes a web style REST API
- Each registered device resource has a corresponding web API resource
- Refer to the mbed Device Server User Guide

Device Server API – Endpoint Discovery

Device Server Endpoint Discovery URL

```
http://10.10.10.10:8080/my_domain/endpoints
```

Domain Name

- Endpoints can be discovered using the Device Server discovery interface
- Endpoint discovery returns a JSON array of endpoint objects

Device Server API – Endpoint Discovery

```
GET http://10.10.10.10:8080/my_domain/endpoints
```


0: {
name: "MBED-061590140030"
type: "LWM2M test client"
status: "ACTIVE"
}-
1: {
name: "mbed-6230600c000f"
type: "mbed_device"
status: "ACTIVE"
}

Device Server API – Resource Discovery

Device Server Resource Discovery URL

```
http://10.10.10.10:8080/my_domain/endpoints/my_ep_name
```

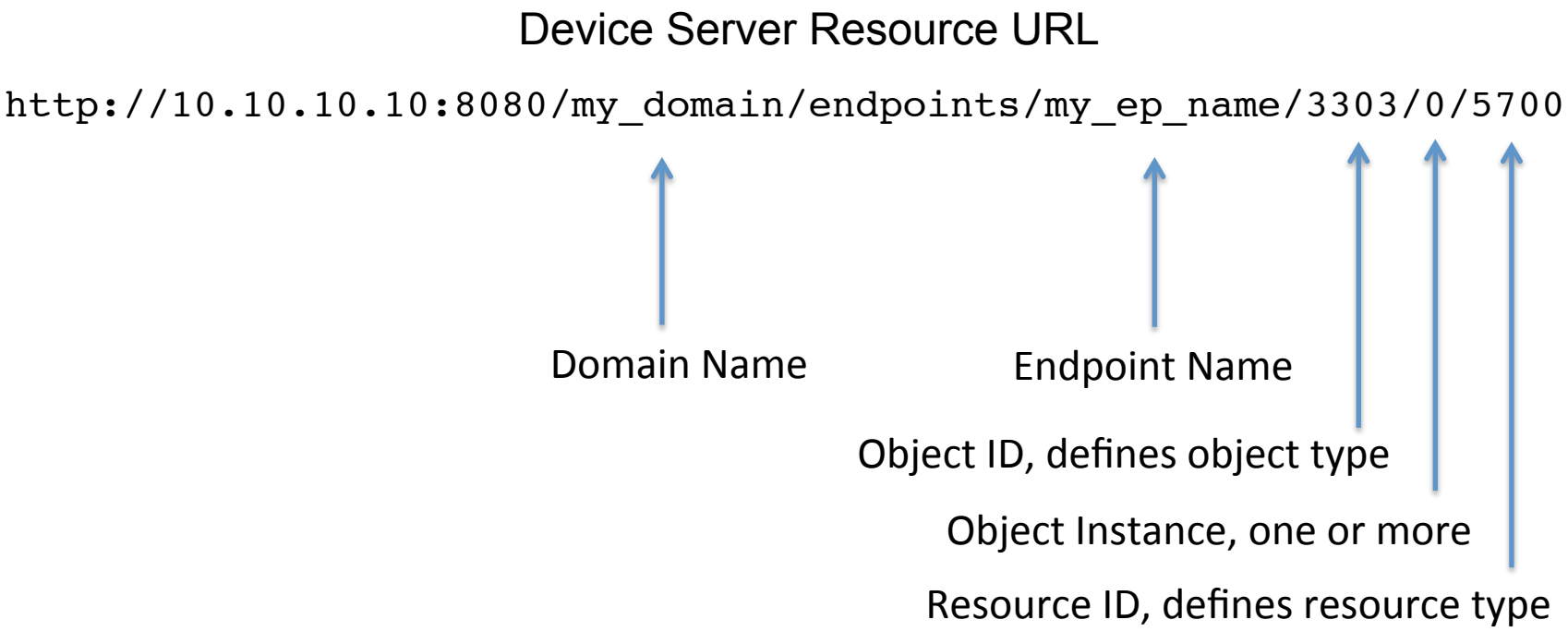
- Resources can be discovered using the Device Server discovery interface
- Resource discovery returns a JSON array of resource objects

Device Server API – Resource Discovery

```
GET http://10.10.10.10:8080/my_domain/endpoints/my_ep_name
```

```
0: {
uri: "/3304/0/5700"
rt: "urn:X-ipso:humidity"
obs: false
type: ""
}-
1: {
uri: "/3303/0/5700"
rt: "urn:X-ipso:temperature"
obs: false
type: ""
}-
2: {
uri: "/3302/0/5500"
rt: "urn:X-ipso:presence"
obs: true
type: ""
}-
```

Device Server API – Resource URL



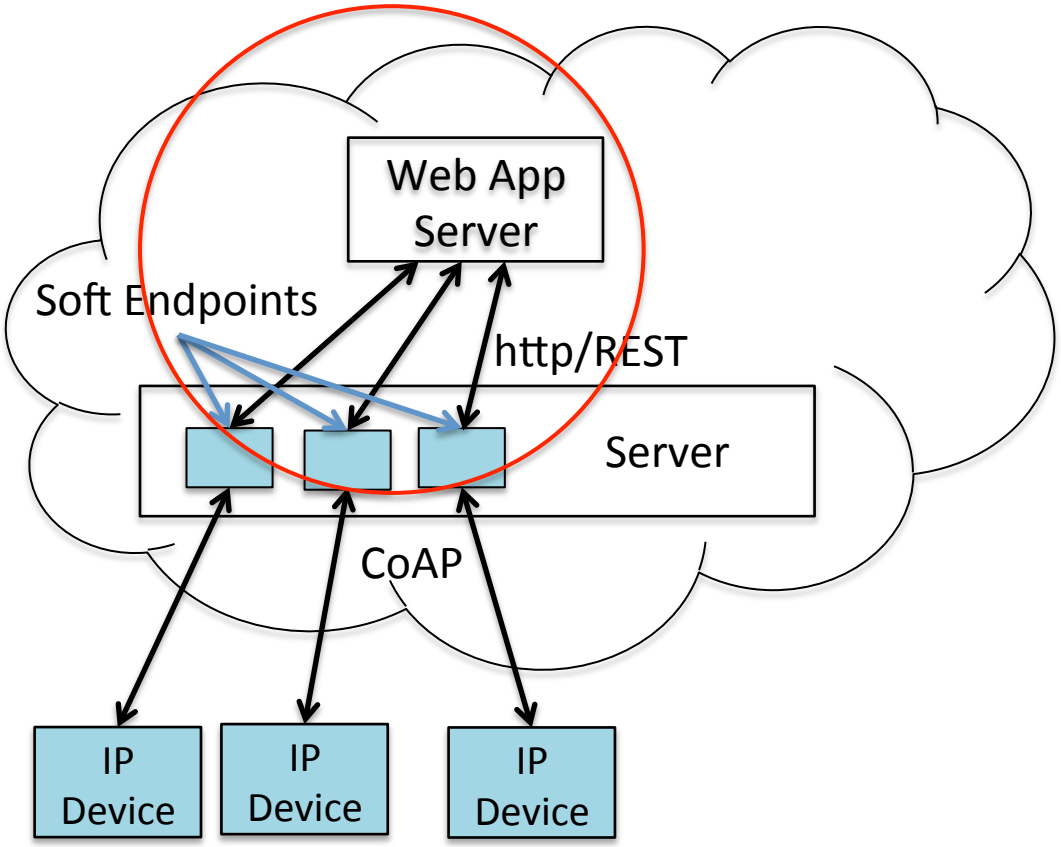
- Web API URLs are constructed from the resource URIs registered by the device, the endpoint name of the registered device, and a preconfigured domain name

Device Server API – Resource URL

```
GET http://10.10.10.10:8080/my_domain/endpoints/my_ep_name/3303/0/5700
```

Returns -> 71.2

Web Application Server



Web Application Server

- The Device Server has REST interfaces for resource discovery, resource access, and asynchronous notification
- Web Application Server runs an application that interacts with the device through the Device Server interfaces
- Application server can run on it's own instance or node, and connect to the device server over the network
- Applications can be programmed in web-friendly languages like Python, Nodejs, Ruby
- This example will focus on nodejs and Node-RED

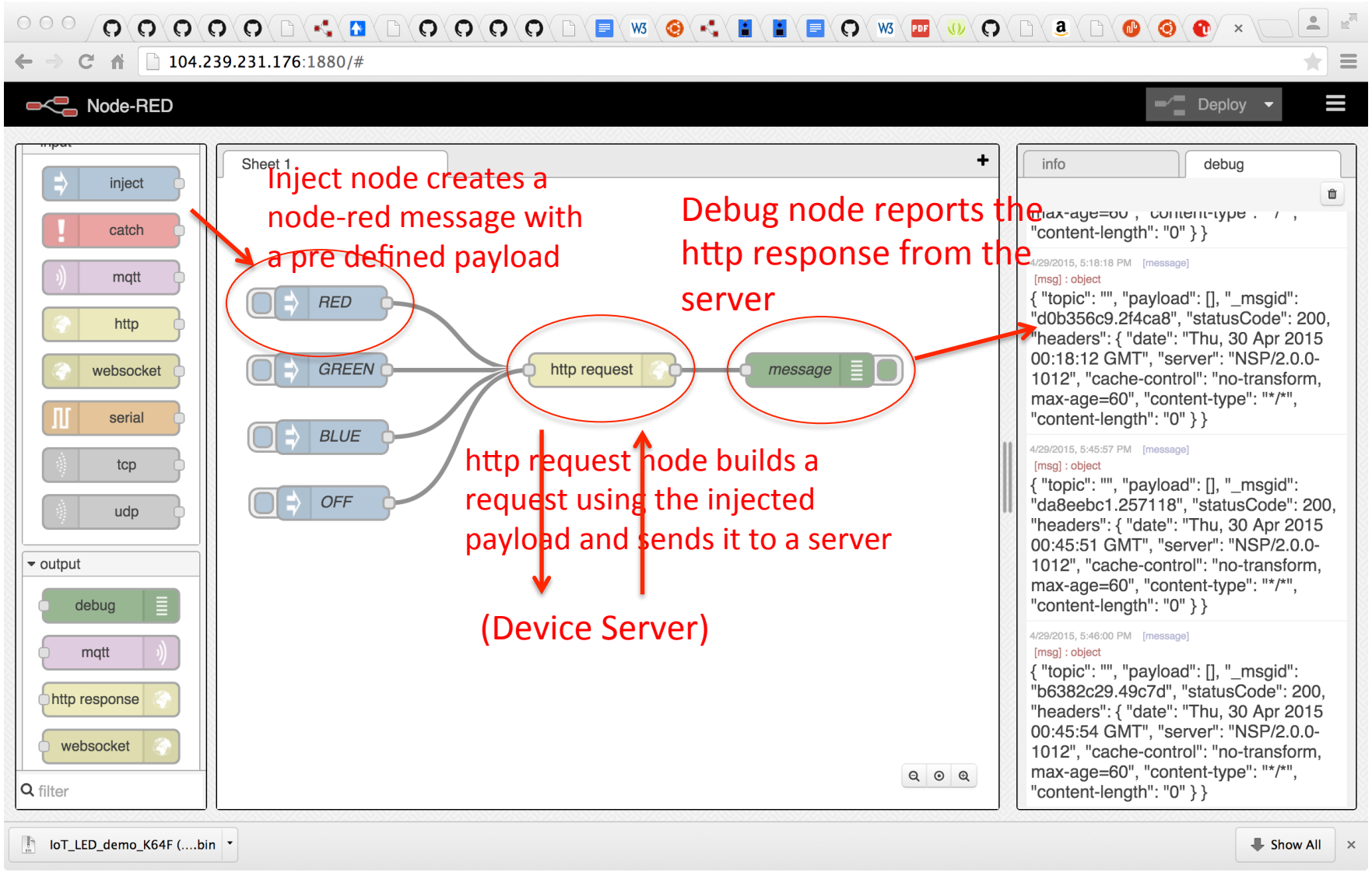
Application Server Setup

- Each team will have a dedicated application server instance available – see the lab notes for details
- Sign up on the Rackspace “iwantaserver” page and get your server’s IP address and root login
- Follow the instructions in the lab note for setting up the server environment and Node-RED
- Test the demo application with your mbed board running the LED Demo project

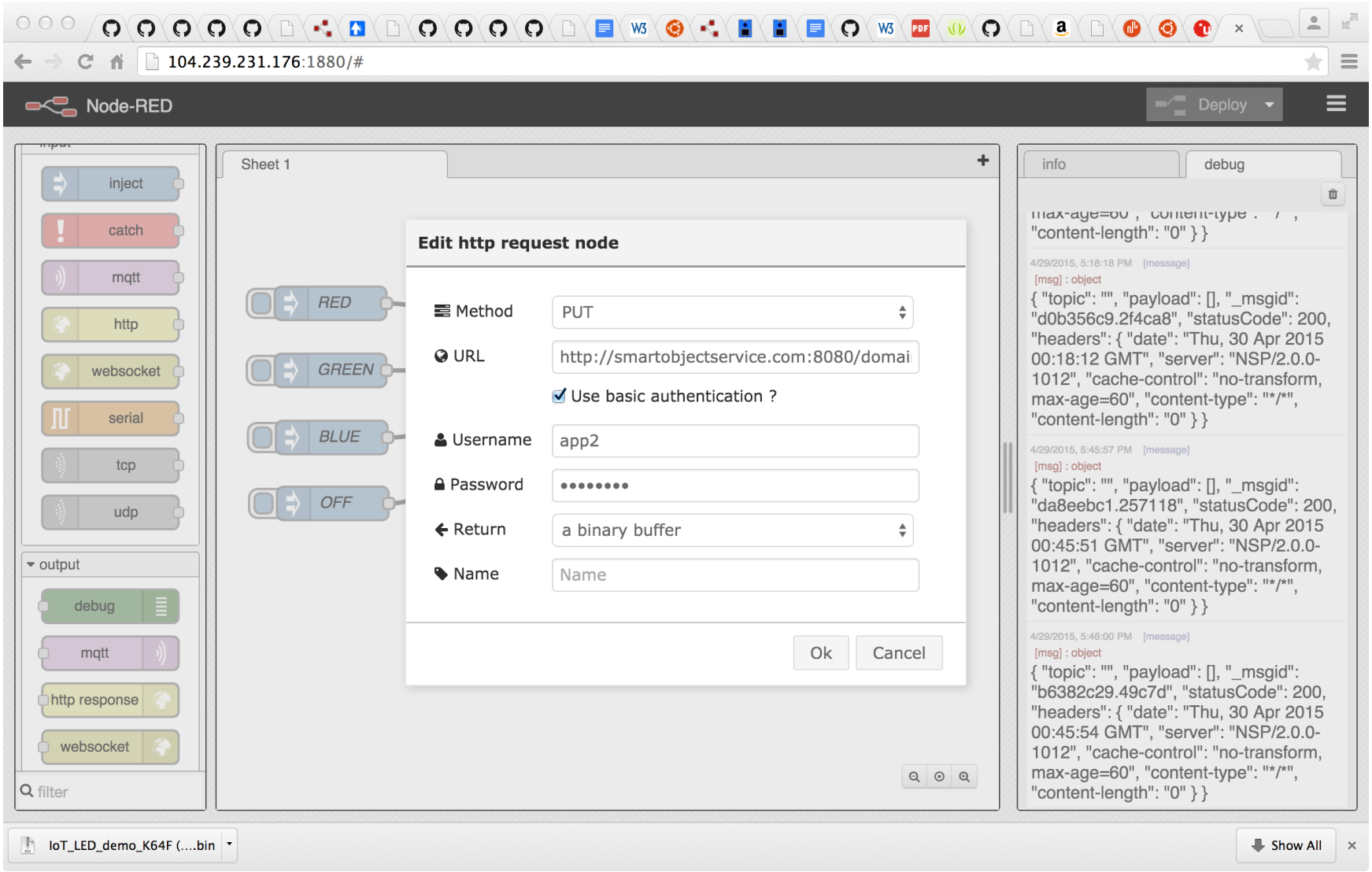
Node-RED

- Node-RED is a visual programming system based on node.js
- Drag and drop programming using pre-configured and custom function nodes
- Event driven data flow programming paradigm with MQTT-style messages consisting of a topic and a payload
- Built-in nodes for HTTP, MQTT, JSON templates, and many other useful functions
- A Node-RED Flow is a collection of nodes linked together

Node-RED Demo Example Flow



Node-RED Configuration



References

IPSO Smart Object Guideline

http://www.ipso-alliance.org/technical-information/ipso-guidelines

Node-RED

http://nodered.org/

mbed Device Server API reference (User Guide)

https://github.com/connectIOT/SpringIotCourse/blob/master/Documents/NanoService_Platform_UserGuide.pdf

Some live examples of mbed Device Server and Node-RED, also MQTT

http://smartobjectservice.com:1880/

Standards References

IPSO Smart Object Guideline

http://www.ipso-alliance.org/technical-information/ipso-guidelines

OMA LWM2M Specification

http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma

IETF CoAP and Related Specifications

CoAP (RFC 7252):
http://tools.ietf.org/html/rfc7252

CoRE Link-Format (RFC 6690):
http://tools.ietf.org/html/rfc6690

CoRE Resource Directory:
http://tools.ietf.org/html/draft-ietf-core-resource-directory-01

CoAP Community Site

http://coap.technology/