OPEN
CONNECTIVITY
FOUNDATION™

IoTivity Architecture

Ashok Subash

Samsung Electronics R & D Institute Bangalore

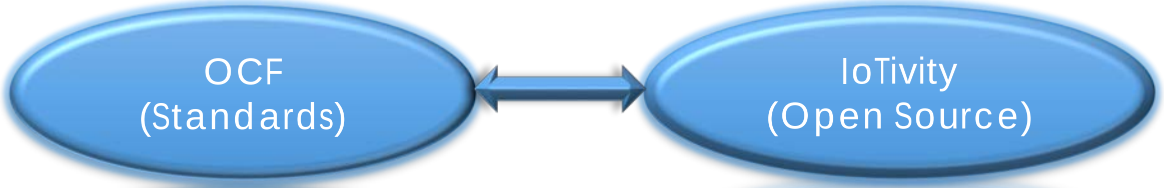
IoTivity

Agenda

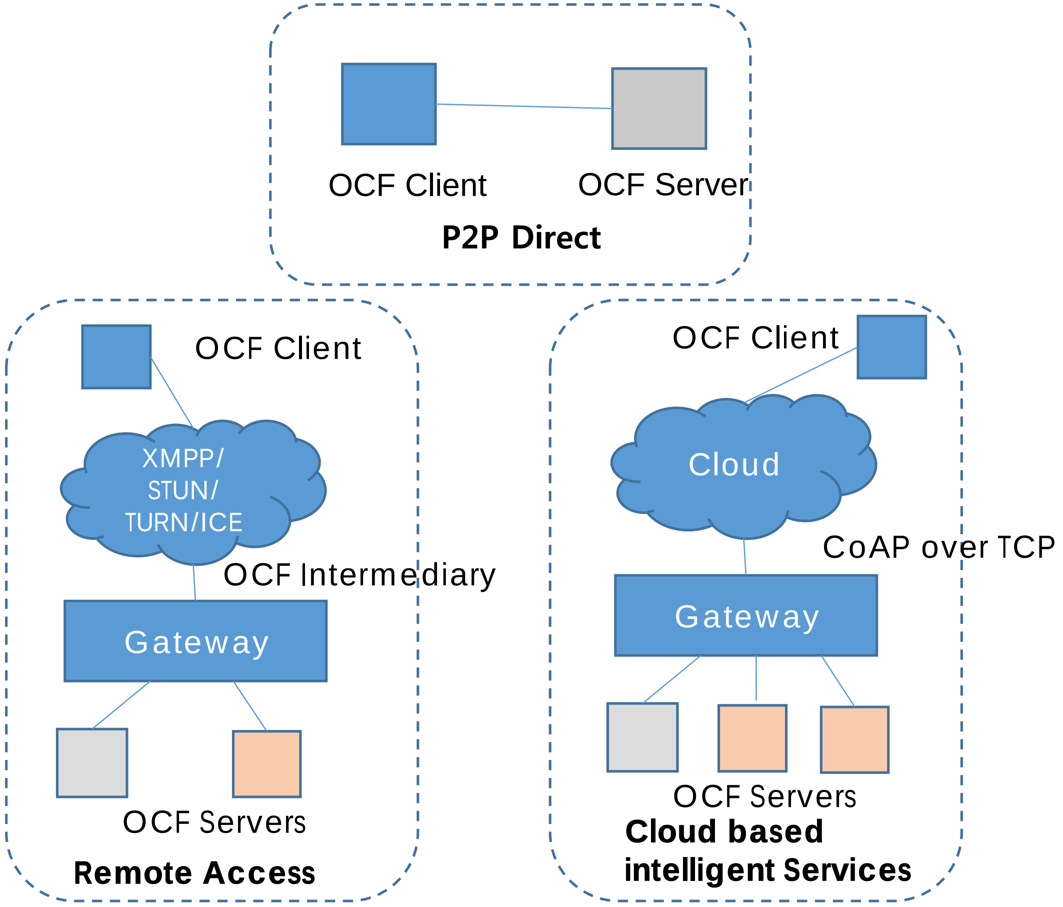
- IoTivity Overview
- IoTivity Architecture
- IoTivity Base Layer & APIs
- IoTivity Primitive Services & APIs
- IoTivity Roadmap

IoTivity Overview

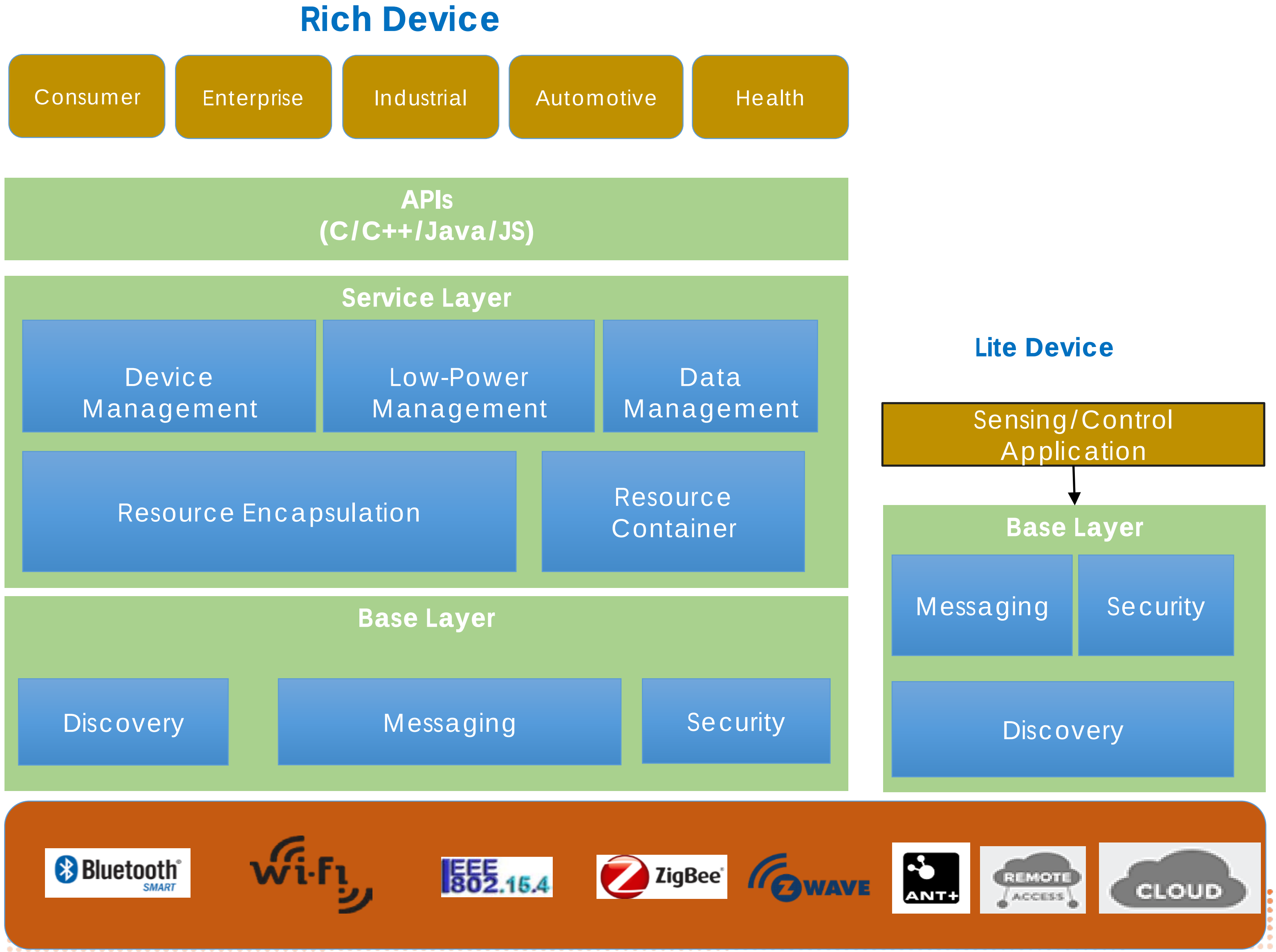
- An open source software framework implementing OCF Standards
- Ensures seamless device-to-device connectivity to address emerging needs of IoT
- Licensed under Apache License Version 2.0
- Available on TIZEN, Android, Arduino, Linux(Ubuntu) Platforms



OCF Topologies Supported



IoTivity – High Level Architecture



Rich Device

Consumer | Enterprise | Industrial | Automotive | Health

APIs (C/C++/Java/JS)

Service Layer

Device Management | Low-Power Management | Data Management

Resource Encapsulation | Resource Container

Base Layer

Discovery | Messaging | Security

Lite Device

Sensing/Control Application

Base Layer

Messaging | Security

Discovery

Key Goals

- Common Solution
- Established Protocols
- Security & Identity
- Standardized Profiles
- Interoperability
- Innovation Opportunities
- Necessary connectivity

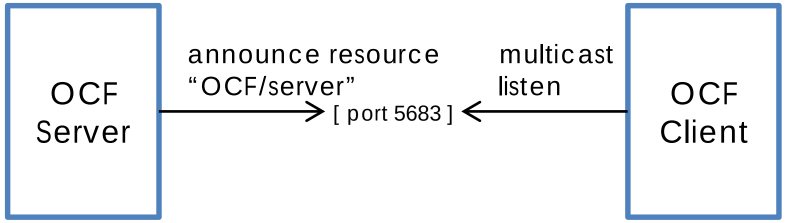
🟨 IoTivity Profiles

🟩 IoTivity Framework

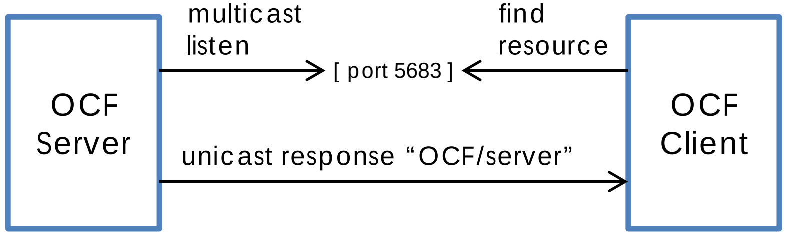
🟧 IoTivity Connectivities*

IoTivity Base Layer & APIs

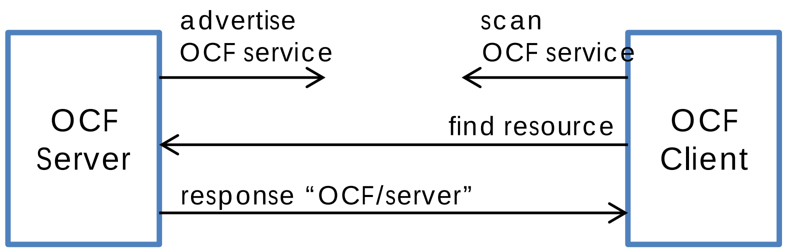
Discovery Subsystem



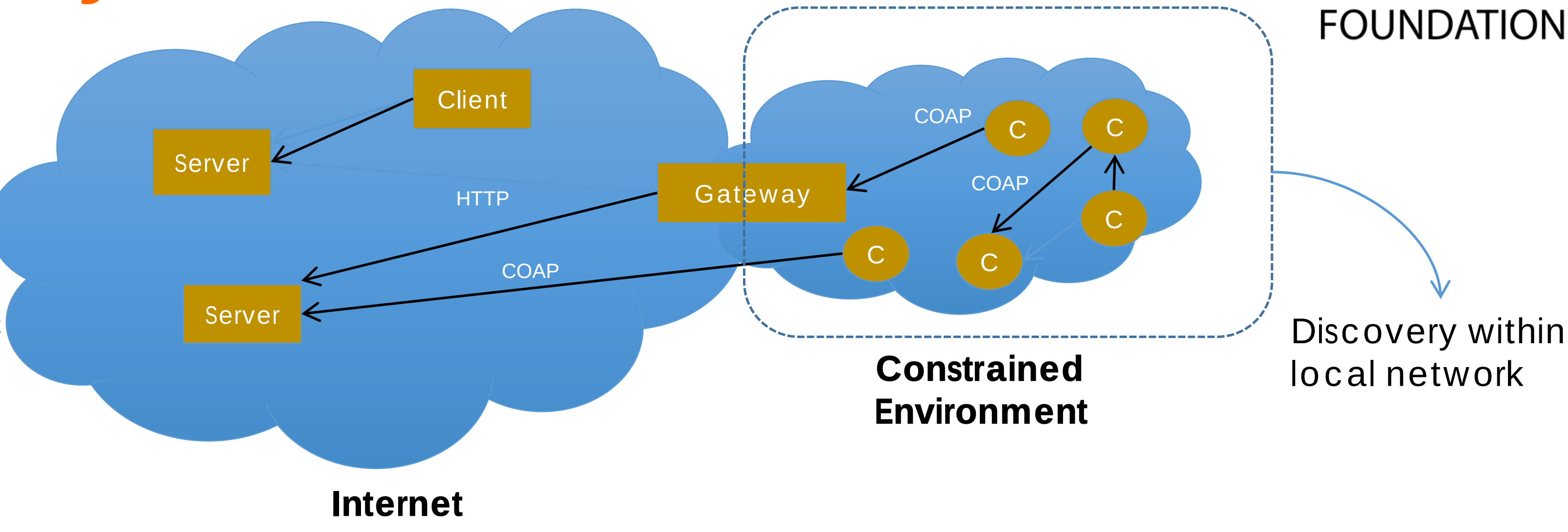
[Figure 1] Multicast announcement over Wi-Fi / Ethernet



[Figure 2] Multicast/Unicast over WiFi / Ethernet



[Figure 3] Advertise/Scan over BLE/BT



Connectivity	Discovery Mechanism	Description
WiFi & Ethernet (over IP)	IP Multicast	CoAP Multicast Port: 5683 (Assigned by IANA) CoAP Secure Port: 5684
	IP Unicast over UDP	Precondition: OIC Server Address & Port are known
Bluetooth (EDR & BLE)	Using Scan & Advertise	OCF Specific Service UUID

CoAP

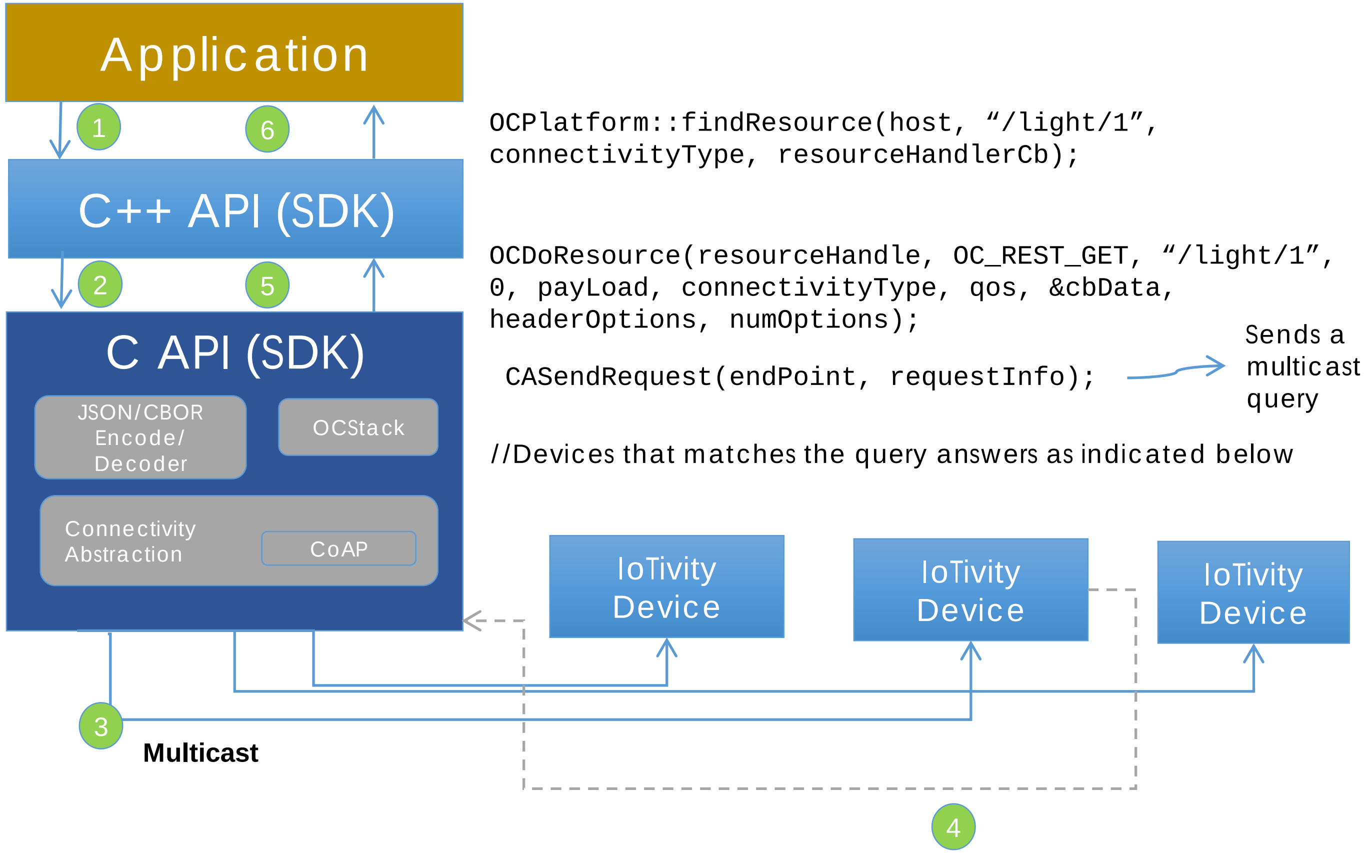
- Open IETF Standard (RFC 7252)
- Compact 4 Byte Header
- UDP (Default), SMS, TCP Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-In Discovery

CoAP: Constrained Application Protocol
IANA: Internet Assigned Numbers Authority

Discovery – Finding a Resource

OPEN
CONNECTIVITY
FOUNDATION™

Function Call Flow



```
OCPlatform::findResource(host, "/light/1",
connectivityType, resourceHandlerCb);
```

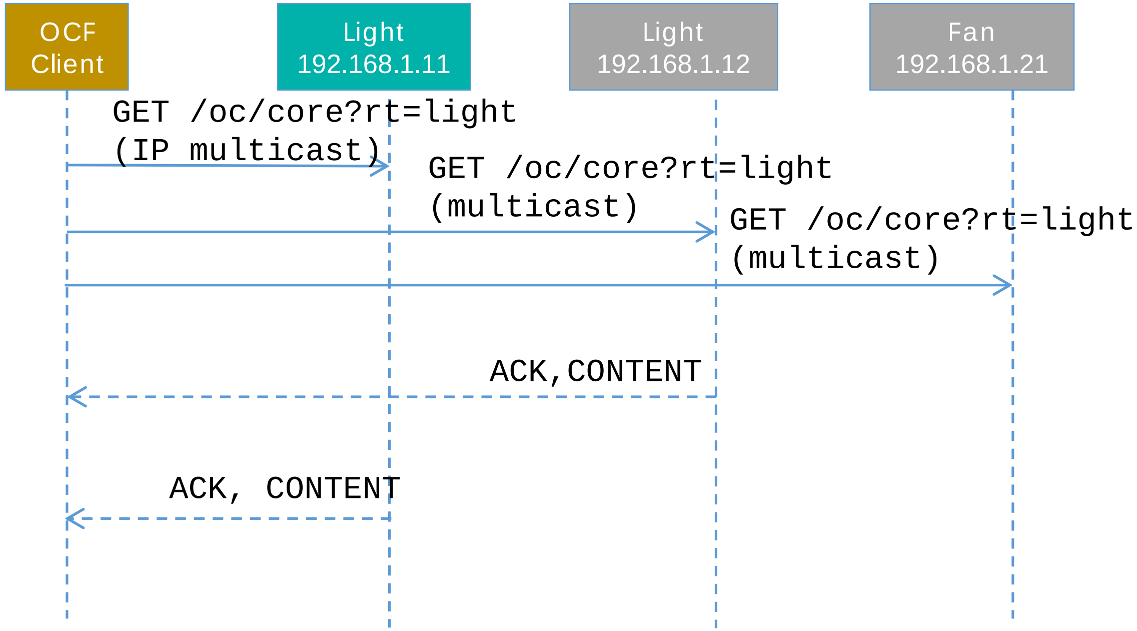
```
OCDoResource(resourceHandle, OC_REST_GET, "/light/1",
0, payLoad, connectivityType, qos, &cbData,
headerOptions, numOptions);
```

```
 CASendRequest(endPoint, requestInfo);
```

Sends a multicast query

//Devices that matches the query answers as indicated below

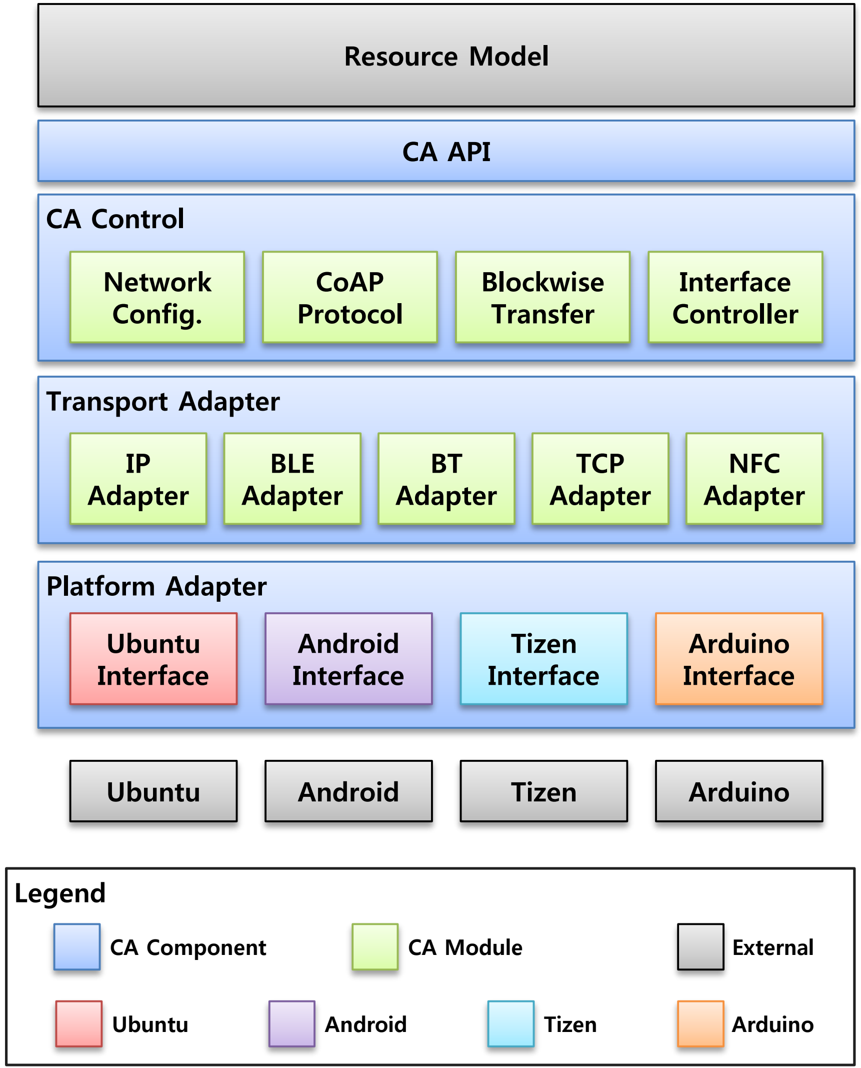
Sequence Diagram



OCF Client
Light 192.168.1.11
Light 192.168.1.12
Fan 192.168.1.21
GET /oc/core?rt=light (IP multicast)
GET /oc/core?rt=light (multicast)
GET /oc/core?rt=light (multicast)
ACK,CONTENT
ACK, CONTENT

IoTivity

Messaging - Connectivity Abstraction



CA Control Component

- Target network selection, interface control & monitoring
- CoAP message serialization & parsing
- Block-wise messaging flow control

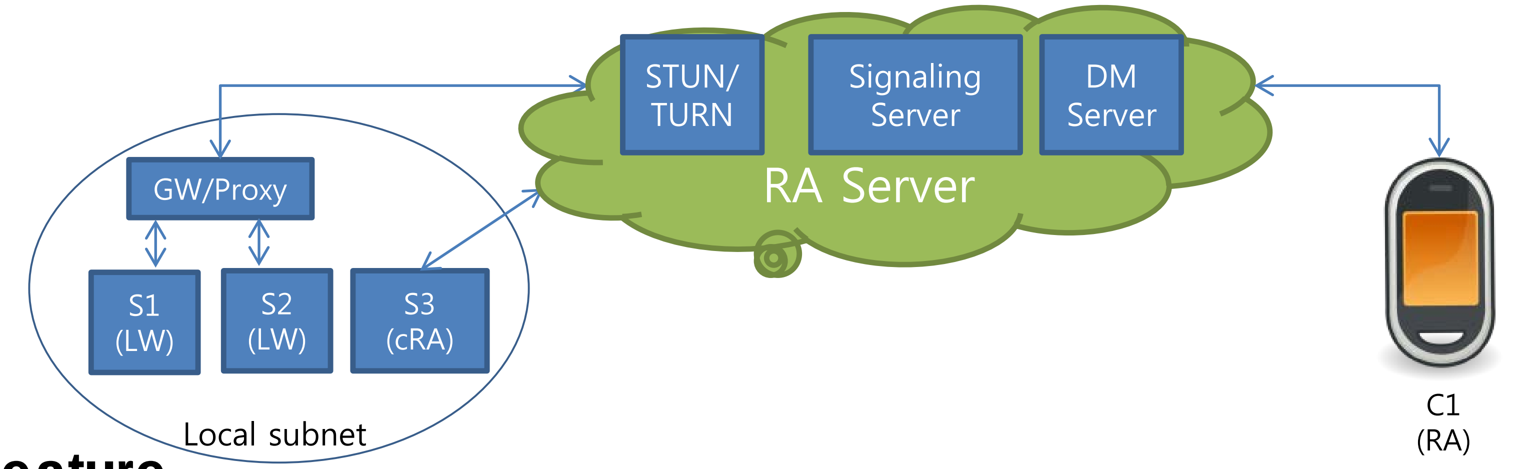
Transport Adapter Component

- Data transmission over UDP, TCP, BLE(GATT), BT(SPP) & NFC
- Secure data exchanging using DTLS

Platform Adapter Component

- Wi-Fi, Ethernet and BLE
- Android Wi-Fi, BLE and BT
- Tizen Wi-Fi, BLE and BT
- Arduino Wi-Fi, Ethernet and BLE

Messaging - Remote Access over XMPP



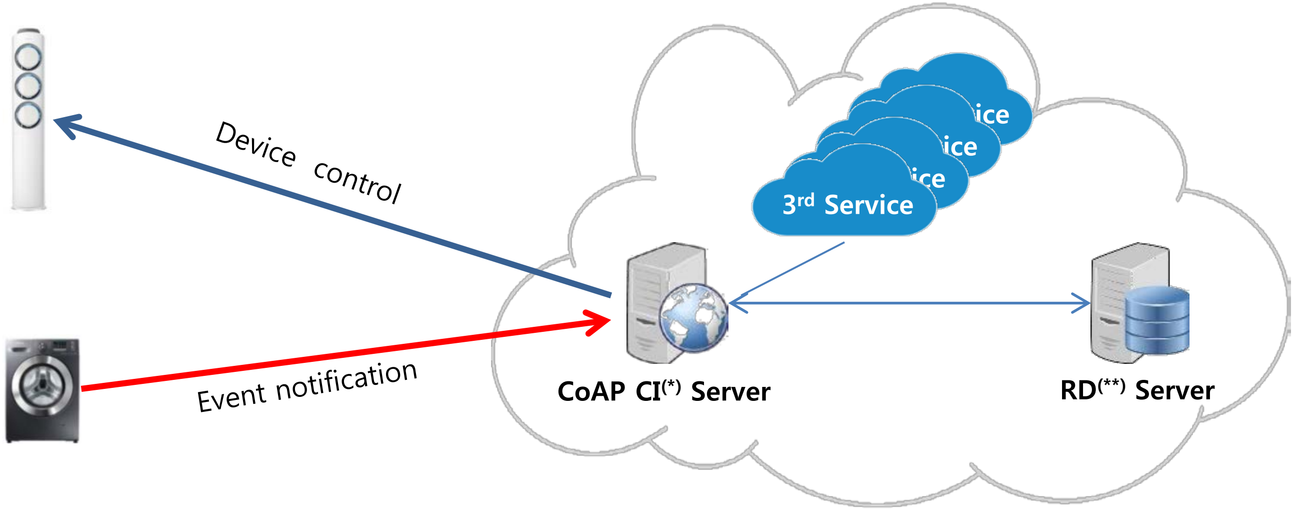
Feature

- Remote client discover & securely interface with resource servers when not on same subnet
- Adheres to access control policies
- End-to-End Secure

Device Type	Use Case
Light weight (LW) Device	Accessible within subnet. No RA, require GW/proxy device for access
Constrained RA (cRA) Endpoint	RA access for non latency-sensitive, low BW applications
RA Endpoint (RA)	Full RA access

Messaging – CoAP over TCP

OPEN CONNECTIVITY FOUNDATION™



CoAP over TCP for Cloud extension

■ TCP and TLS Transport for the CoAP

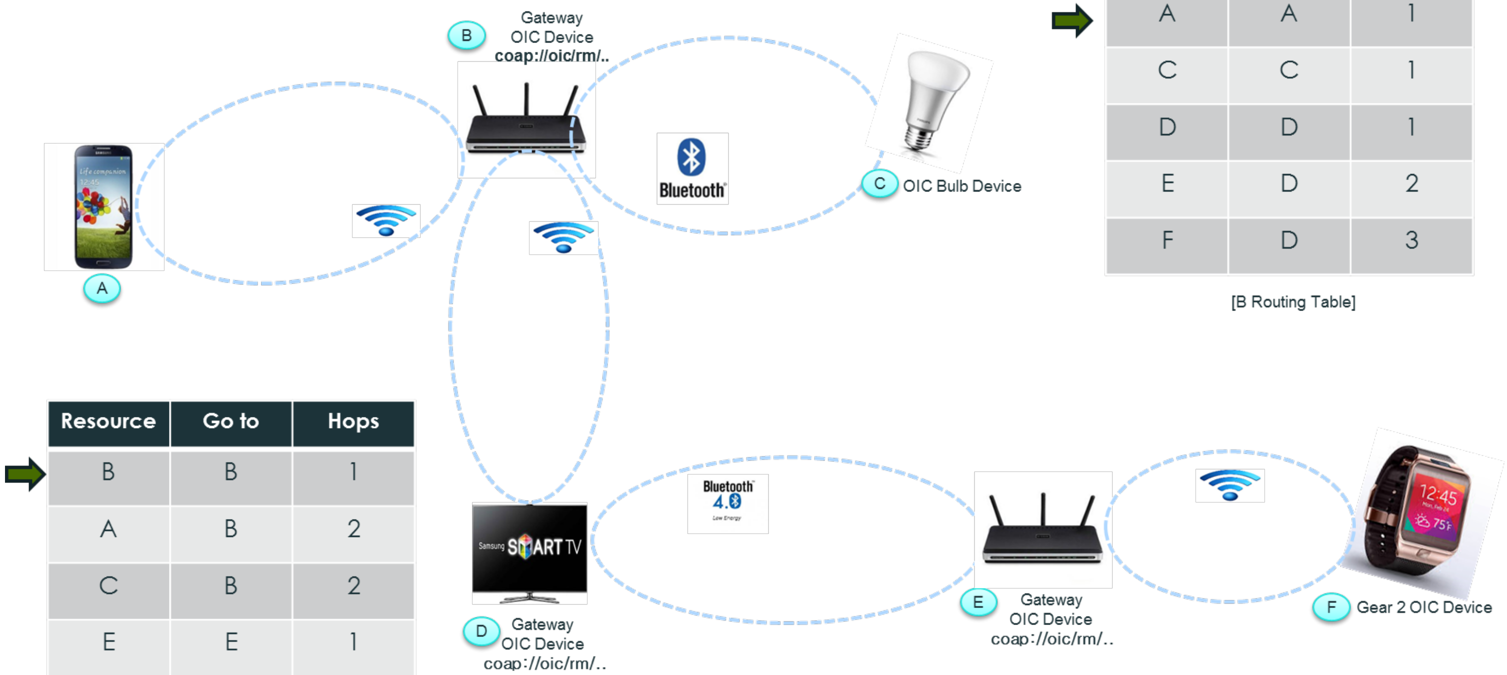
- ❖ CoAP Default transport - UDP.
  - Reliable delivery, simple congestion control & flow control
  - Provided by the message layer of CoAP
- ❖ CoAP over TCP Benefits .
  - To integrate well with existing enterprise infrastructure,
  - Ability to work with existing NAT boxes
  - Advanced Congestion Control algorithms
  - Integration with Web Environment
- ❖ Resources should be registered to the Resource Directory Service for discovery

* CI : Cloud Interface
** RD : Resource Directory

IoTivity

Message Switching

- To Pass IoTivity messages through heterogeneous network
- Uses DSDV* routing algorithm
- Table-driven routing scheme for ad-hoc mobile network
- Uses CoAP Option



Resource	Go to	Hops
A	A	1
C	C	1
D	D	1
E	D	2
F	D	3

[B Routing Table]

Resource	Go to	Hops
B	B	1
A	B	2
C	B	2
E	E	1
F	E	2

[D Routing Table]

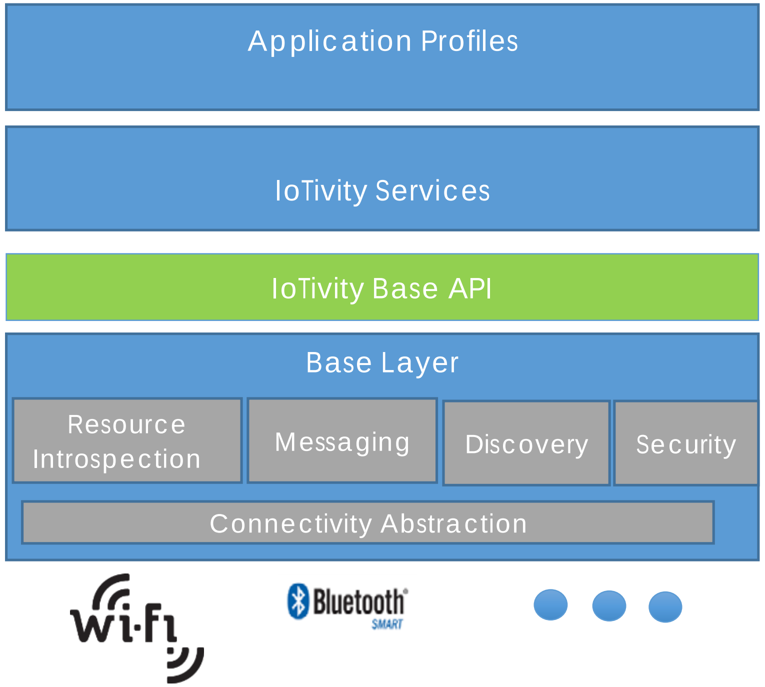
*Destination-Sequenced Distance-Vector Routing

Source Length	Source Address	Destination Length	Destination Address	Multicast Sequence No.
1 Byte	Length specified in Source Length	1 Byte	Length Specified in Destination Length	1 Byte

Programming IoTivity Base APIs

Steps Involved

- Registering a Resource
- Finding a Resource
- Querying a Resource State
- Setting a Resource State
- Observing Resource State



Application Profiles

IoTivity Services

IoTivity Base API

Base Layer

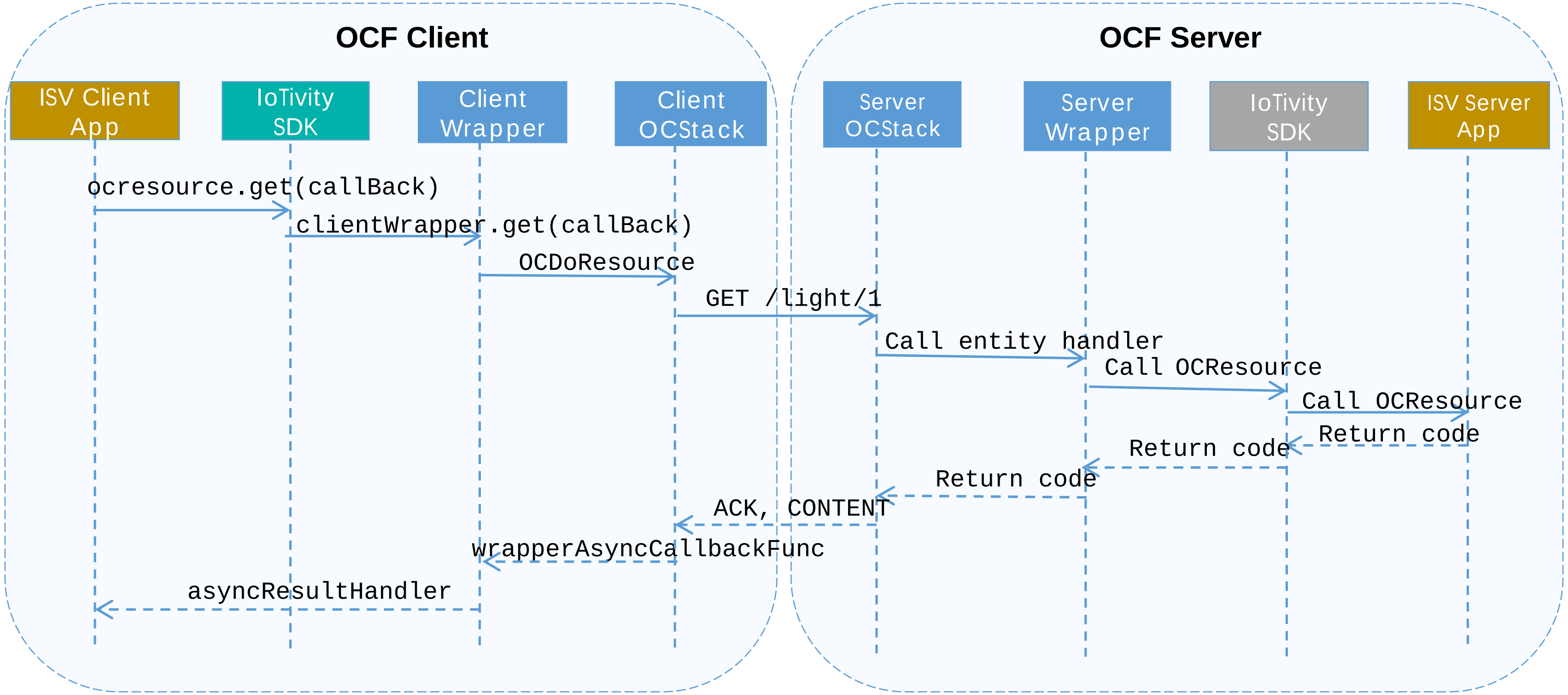
Resource Introspection

Messaging

Discovery

Security

Connectivity Abstraction



OCF Client

OCF Server

ISV Client App

IoTivity SDK

Client Wrapper

Client OCStack

Server OCStack

Server Wrapper

IoTivity SDK

ISV Server App

ocresource.get(callBack)

clientWrapper.get(callBack)

OCDoResource

GET /light/1

Call entity handler

Call OCResource

Call OCResource

Return code

Return code

Return code

ACK, CONTENT

wrapperAsyncCallbackFunc

asyncResultHandler

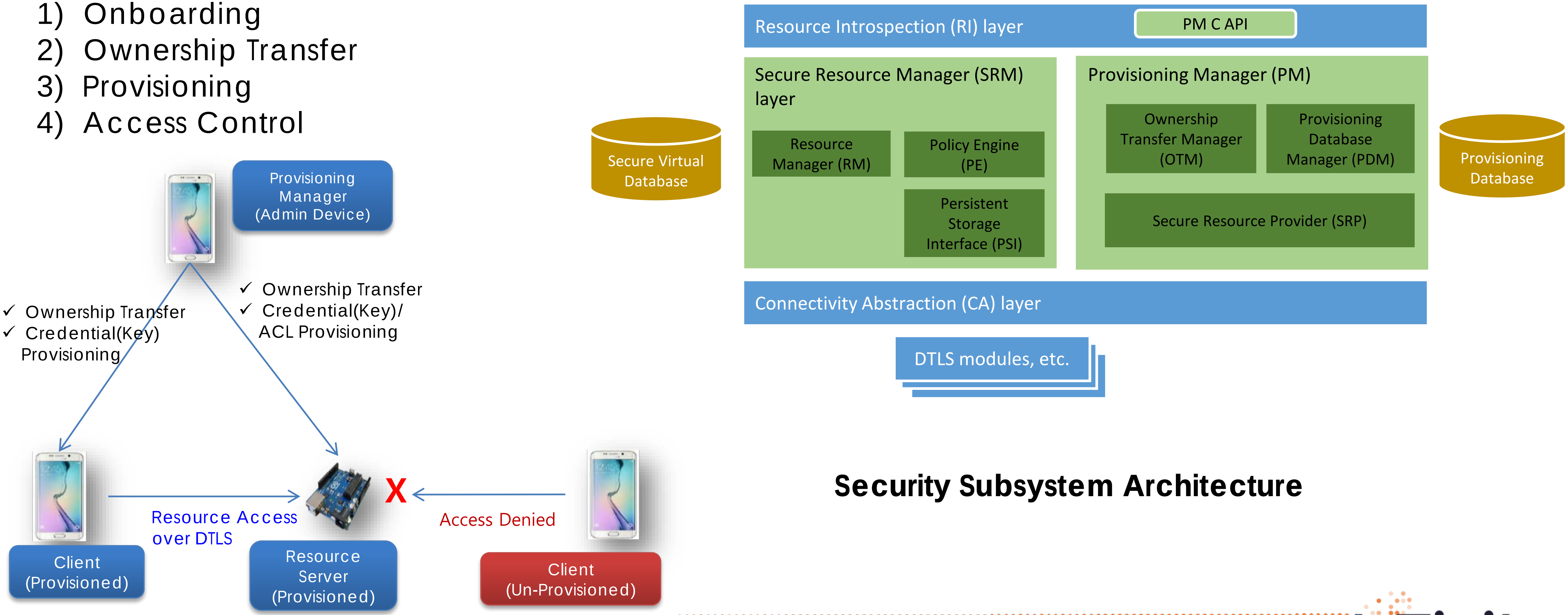
Querying a Resource State: Sequence Diagram

IoTivity Security

Security Features & Architecture

Key Functionality

1) Onboarding
2) Ownership Transfer
3) Provisioning
4) Access Control


Provisioning Manager (Admin Device)
✓ Ownership Transfer
✓ Credential(Key) Provisioning
✓ Ownership Transfer
✓ Credential(Key)/ ACL Provisioning
Client (Provisioned)
Resource Access over DTLS
Resource Server (Provisioned)
X
Access Denied
Client (Un-Provisioned)
Resource Introspection (RI) layer
PM C API
Secure Resource Manager (SRM) layer
Resource Manager (RM)
Policy Engine (PE)
Persistent Storage Interface (PSI)
Provisioning Manager (PM)
Ownership Transfer Manager (OTM)
Provisioning Database Manager (PDM)
Secure Resource Provider (SRP)
Secure Virtual Database
Provisioning Database
Connectivity Abstraction (CA) layer
DTLS modules, etc.

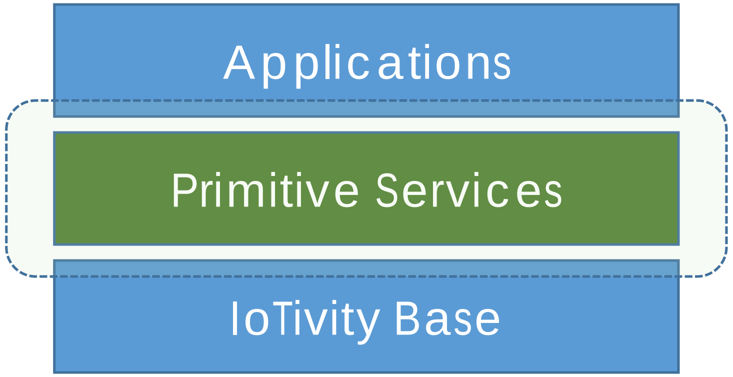
Security Subsystem Architecture

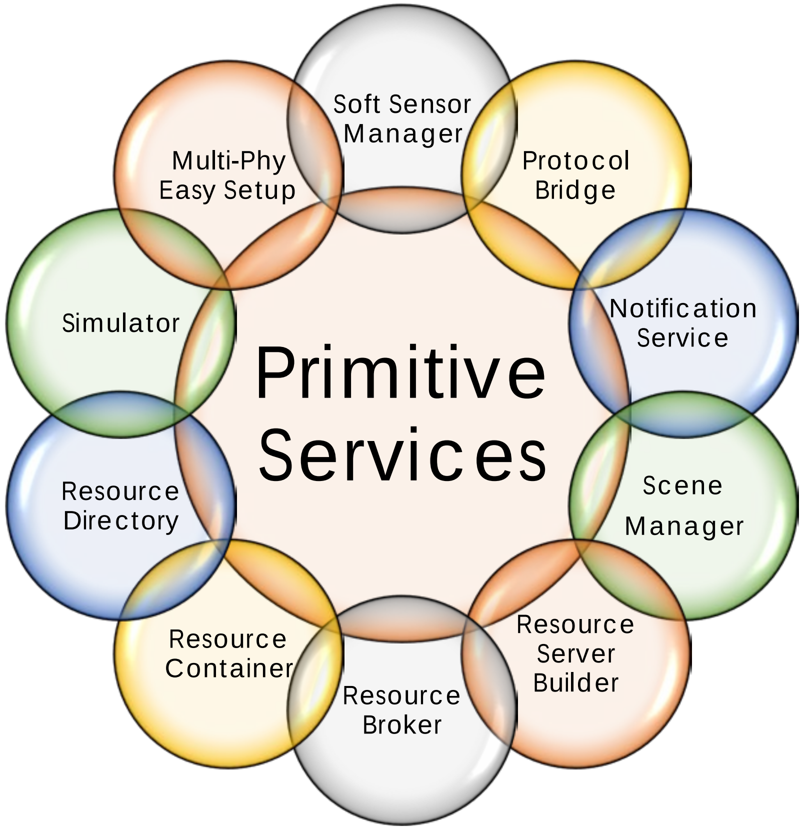
*Platform Hardening not part of the OCF Specs & IoTivity Implementation

IoTivity Primitive Services & APIs

Purpose of Primitive Services

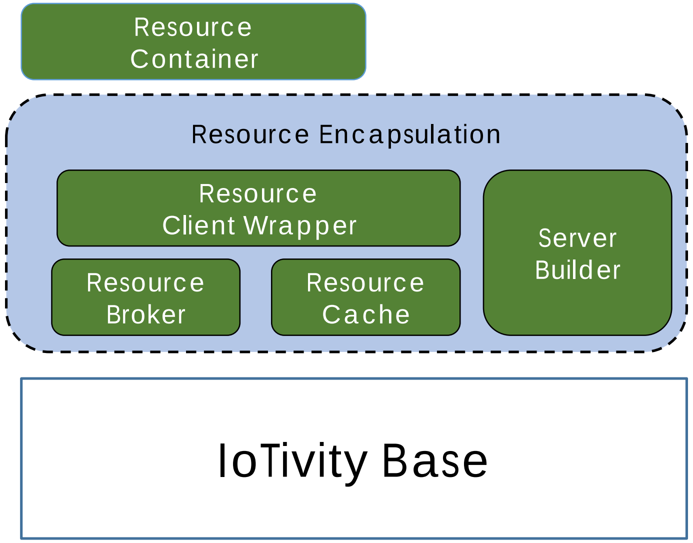
- ❖ Provides easier and simpler APIs for App developers (Heavy Lifting done by Framework)
- ❖ Mostly designed to run on Smart or Controller devices
- ❖ Uses the Iotivity Base APIs





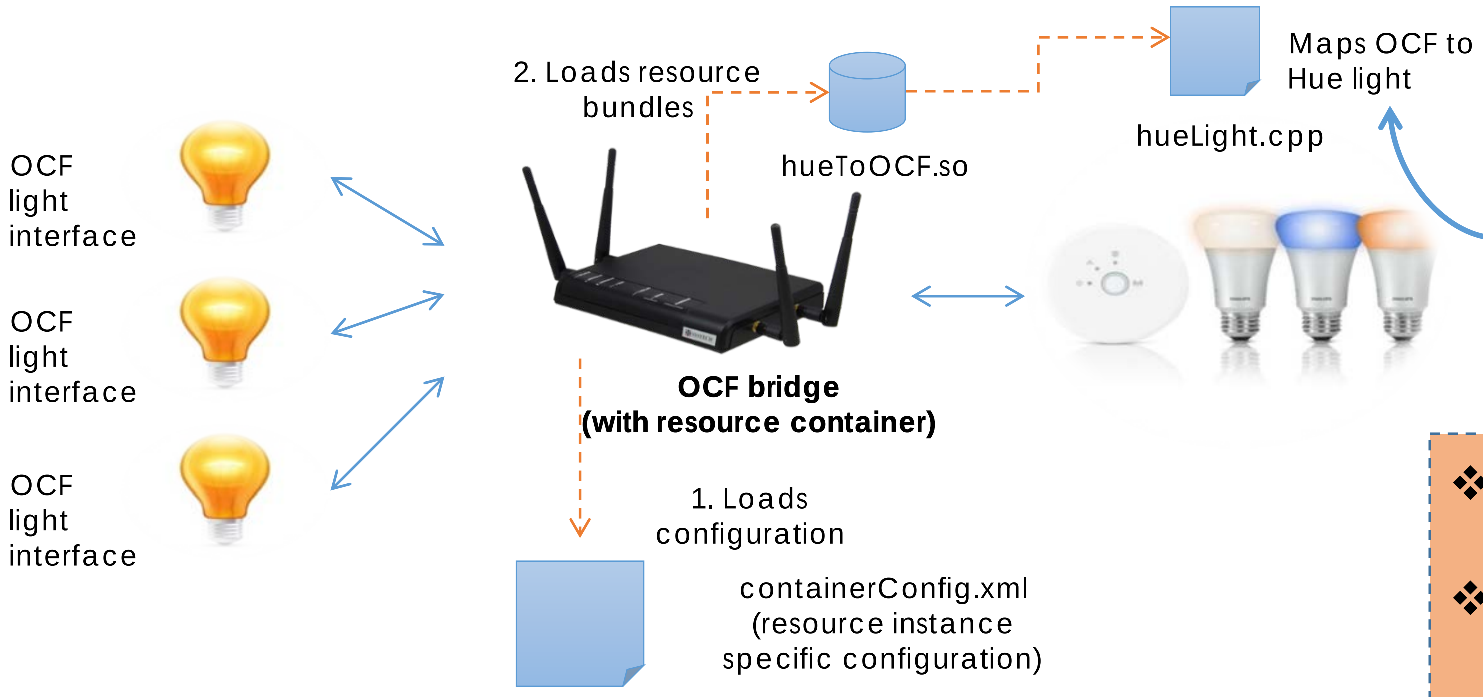
Resource Encapsulation

OPEN CONNECTIVITY FOUNDATION™



Module	Description
Resource Broker	- Remote Resource Presence check (regardless of Remote Server supporting presence feature) - Provide consistent reachability management for discovered resource of interest
Resource Cache	- Maintains last information of Remote Resource (regardless of Remote Server is observable) - Data Centric API (Send/Recv Message Getter/Setter, Data Cache)
Server Builder	- Att. setter to provide easy way to create resource - Changes “msg Handling” to “Data Setting” for users - Monitors value of attributes so that notify-back for observation whenever attribute has changed

Protocol Bridge using Resource Container



OCF light interface

OCF light interface

OCF light interface

2. Loads resource bundles

hueToOCF.so

hueLight.cpp

Maps OCF to Hue light

OCF bridge
(with resource container)

1. Loads configuration

containerConfig.xml
(resource instance specific configuration)

- Integrates non-OCF resources (Bundle)
- Handles dynamic loading of resource bundles & dynamic creation of resources
- Supports C++ .so files & Java .jar files
- Common configuration for bundles and configured resources

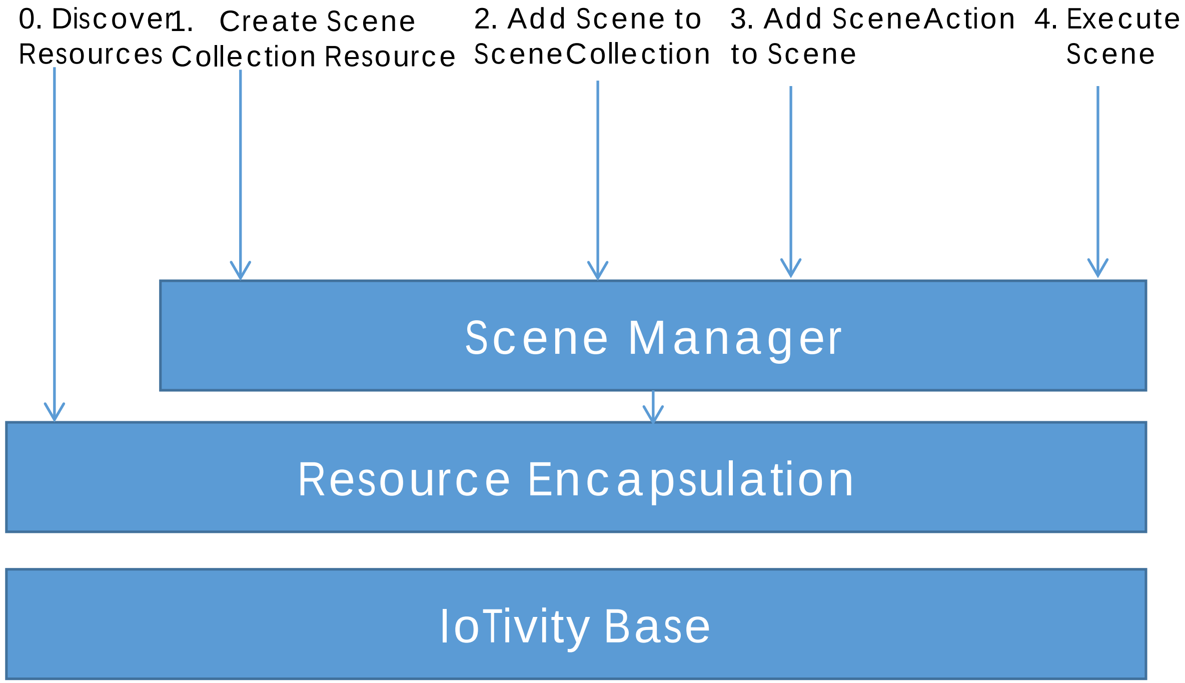
❖ Designed to work devices with non OCF devices

❖ Enables control of legacy devices which are already in market with existing APIs using a OIC complaint device

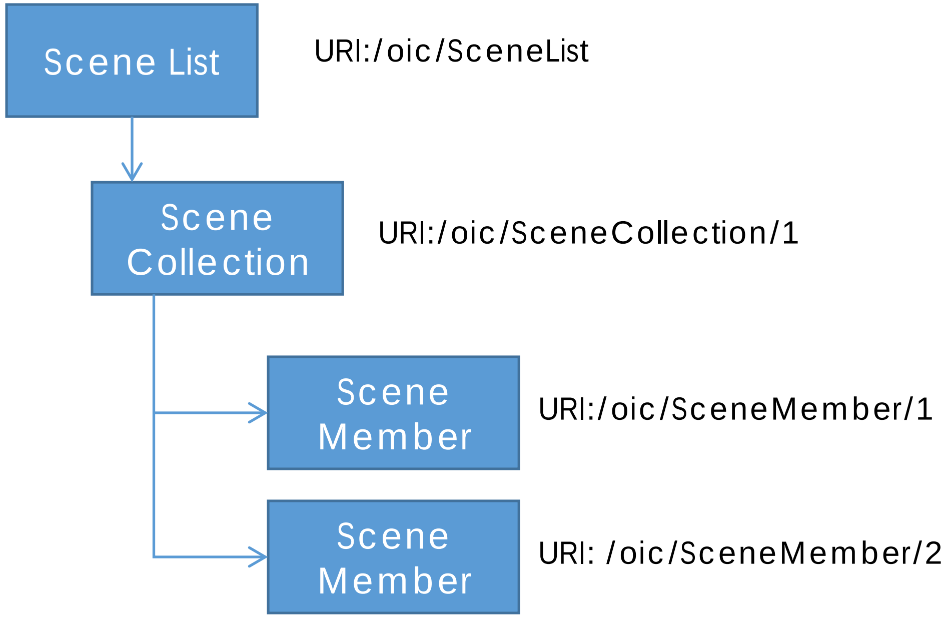
Scene Manager

Helps Users to create a Scenario or Scene for controlling Multiple IoT devices & their functionality

e.g. Away Home – All Lights turned off, Doors locked
Watching Movie – Living Room lights off, TV On, Speaker On



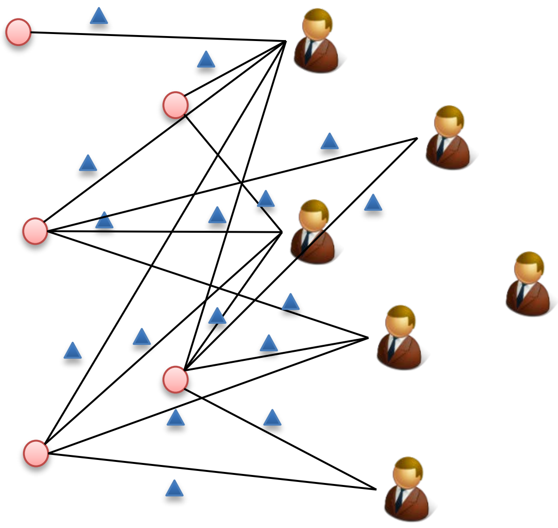
0. Discover Resources
1. Create Scene Collection Resource
2. Add Scene to SceneCollection
3. Add SceneAction to Scene
4. Execute Scene
Scene Manager
Resource Encapsulation
IoTivity Base



Resource Model

Low Power Management – Resource Hosting

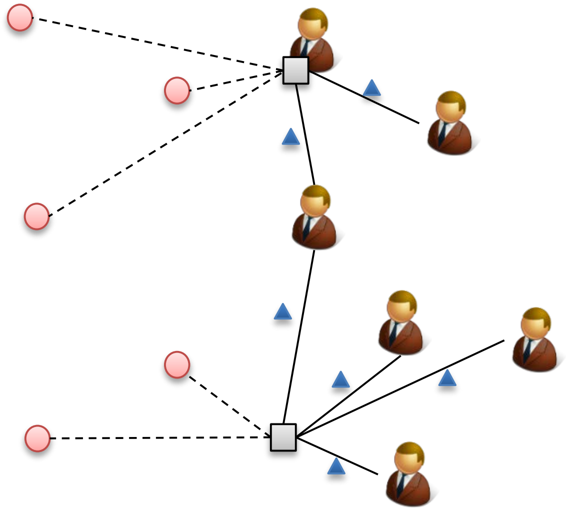
Problem



How many subscriptions thin device could support with its constrained system resource?

○ Thin(Light) device ▲ Subscription 👤 User/Consumer

Solution



Thin Device enhances its lifetime delegating its resource subscriber to richer hosting device

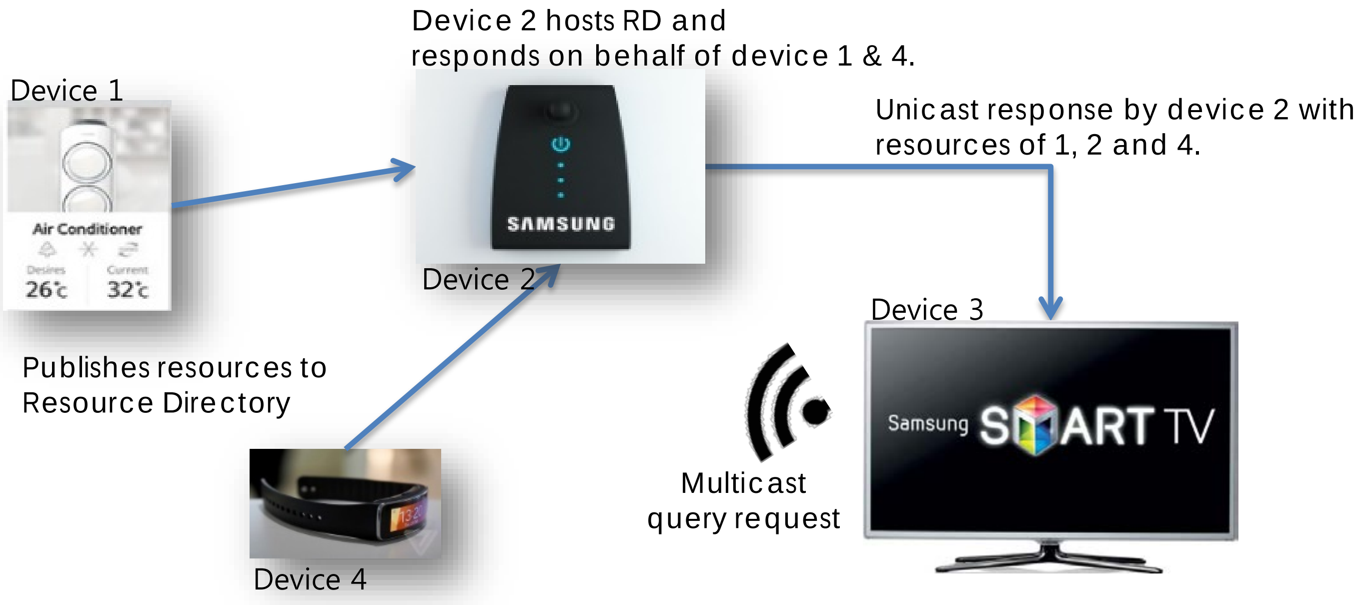
□ Hosting(Rich) device

○ Thin(Light) device

▲ Subscription 👤 User/Consumer

- ❖ Offloads request/data handling from remote clients
- ❖ Reduces the power consumption of resource constraint device

Low Power Management – Resource Directory



Device 2 hosts RD and responds on behalf of device 1 & 4.

Unicast response by device 2 with resources of 1, 2 and 4.

Publishes resources to Resource Directory

Multicast query request

- Constrained device that needs to sleep and can not respond to multicast discovery queries
  - Discovery of RD server
  - Publish Resource to RD
  - Update / Delete Resource

MultiPhy Easy Setup

OPEN CONNECTIVITY FOUNDATION™

- Mediator
  - ✓ E.g., UI-capable Smartphone
- Enrollee
  - ✓ E.g., Out-of-box and UI-less Thing
- Enroller
  - ✓ E.g., WiFi AP, Zigbee Coordinator

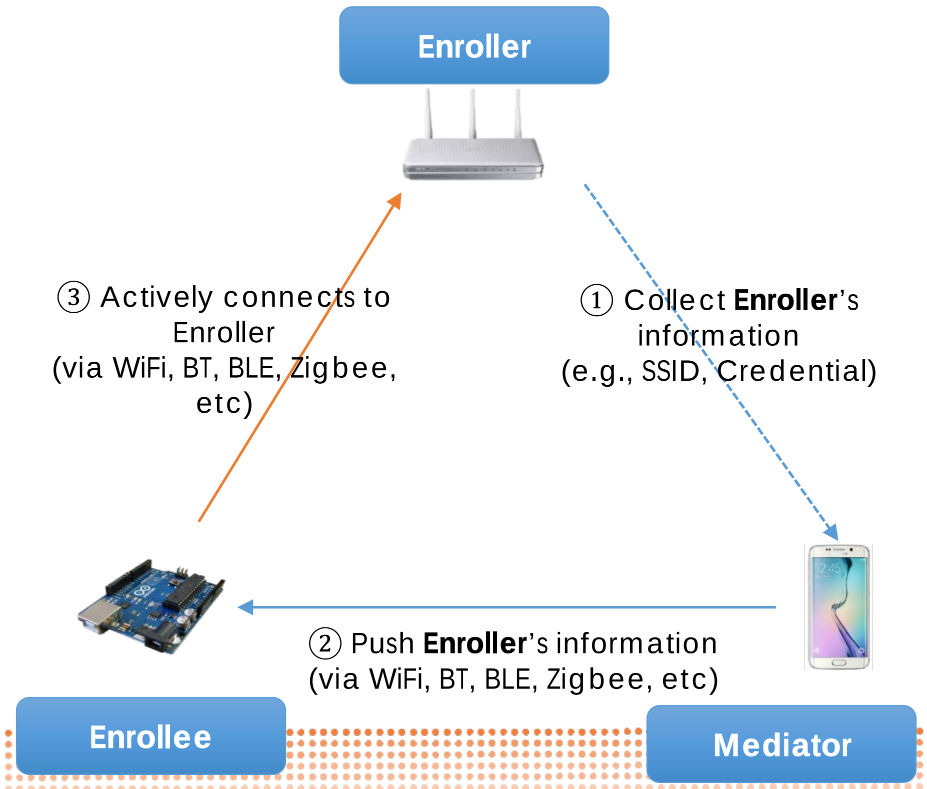
<Mode 1>

Network type (onboarding)	1(WiFi, STA)
SSID	OIC-network
Passphrase type	WPA-PSK2
Passphrase	password
Target network type	3(WiFi, STA)

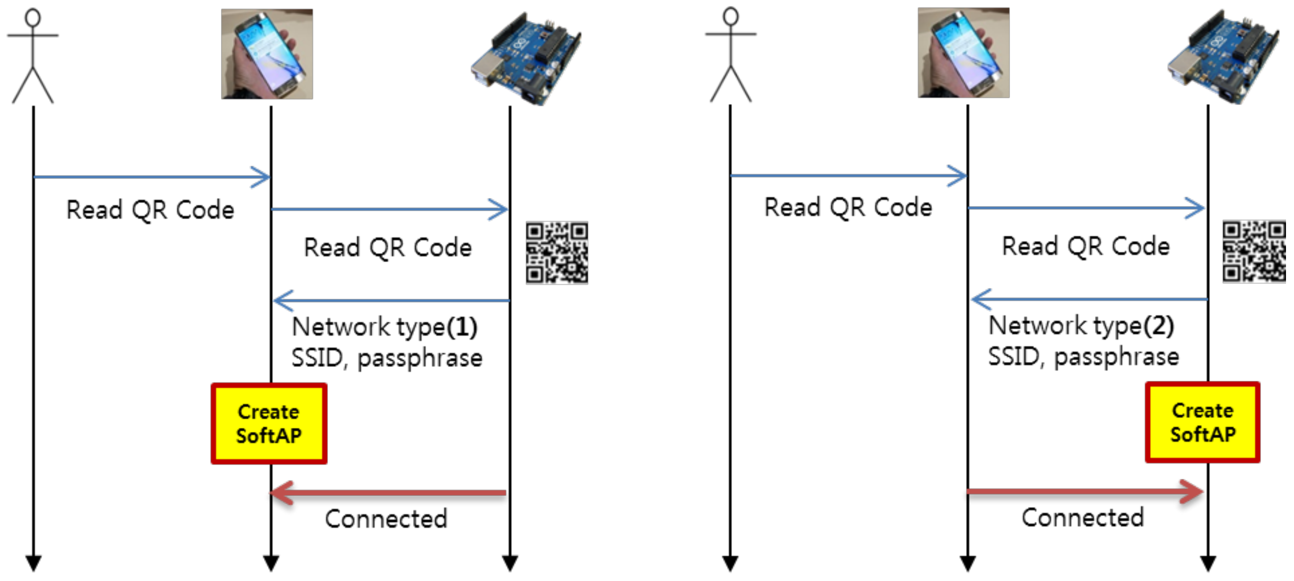
<Mode 2>

Network type (onboarding)	2(WiFi, SoftAP)
SSID	OIC-network
Passphrase type	WPA-PSK2
Passphrase	password
Target network type	3(WiFi, STA)

Scenario



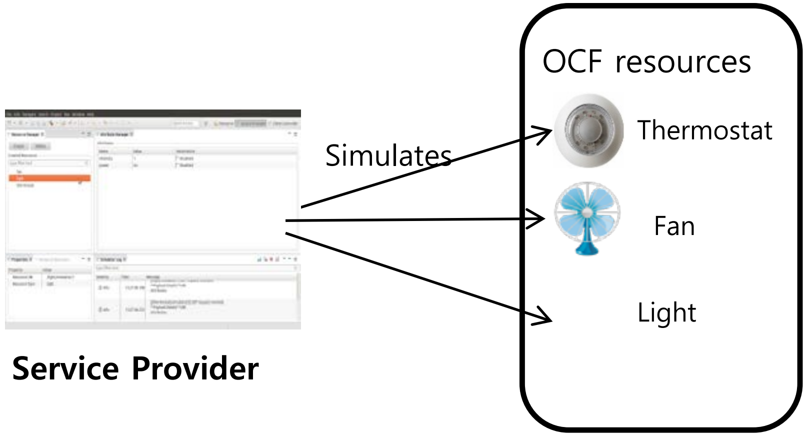
Enroller
③ Actively connects to Enroller (via WiFi, BT, BLE, Zigbee, etc)
① Collect Enroller's information (e.g., SSID, Credential)
② Push Enroller's information (via WiFi, BT, BLE, Zigbee, etc)
Enrollee
Mediator



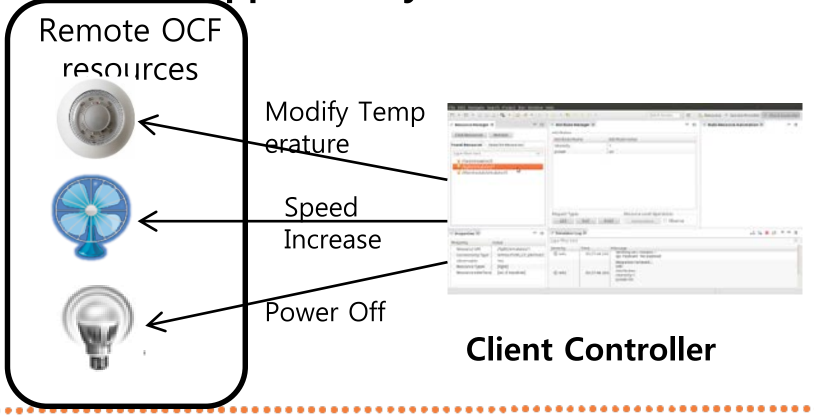
IoTivity

Simulator Service

Simulating different OCF resources



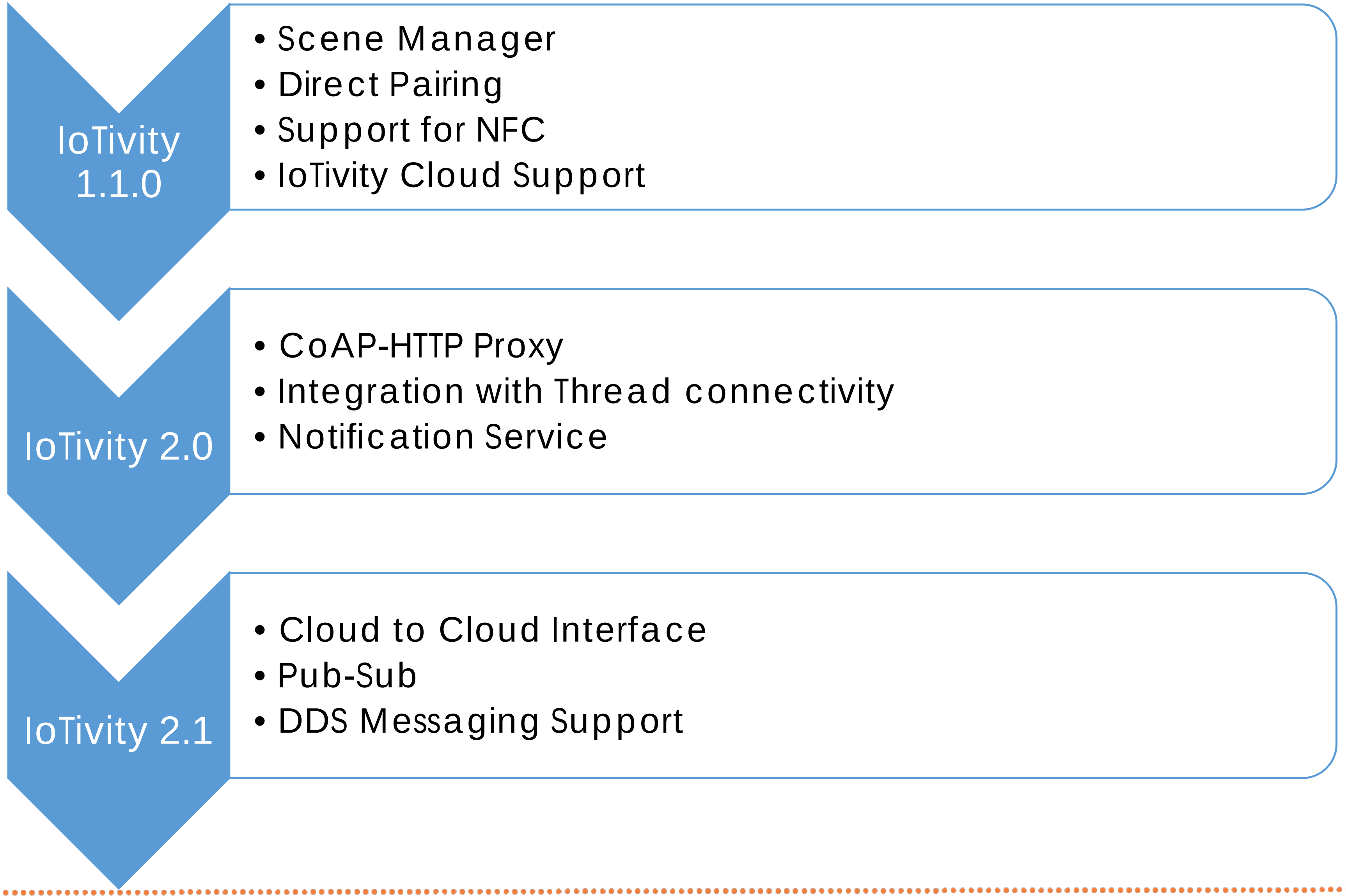
Sending different requests to verify features supported by OCF resources



Feature

- **Server**
  - OCF resources can be simulated, Using resource model definition (RAML) files.
  - Manages creation, deletion, request handling and notifications for OCF resources.
- **Client**
  - Searching for different types of resources available in the network.
  - Sending different types of requests both manual and automatically and displays the response payload received.

IoTivity Roadmap



March 2016

IoTivity 1.1.0

- Scene Manager
- Direct Pairing
- Support for NFC
- IoTivity Cloud Support

Sep 2016

IoTivity 2.0

- CoAP-HTTP Proxy
- Integration with Thread connectivity
- Notification Service

Not finalized

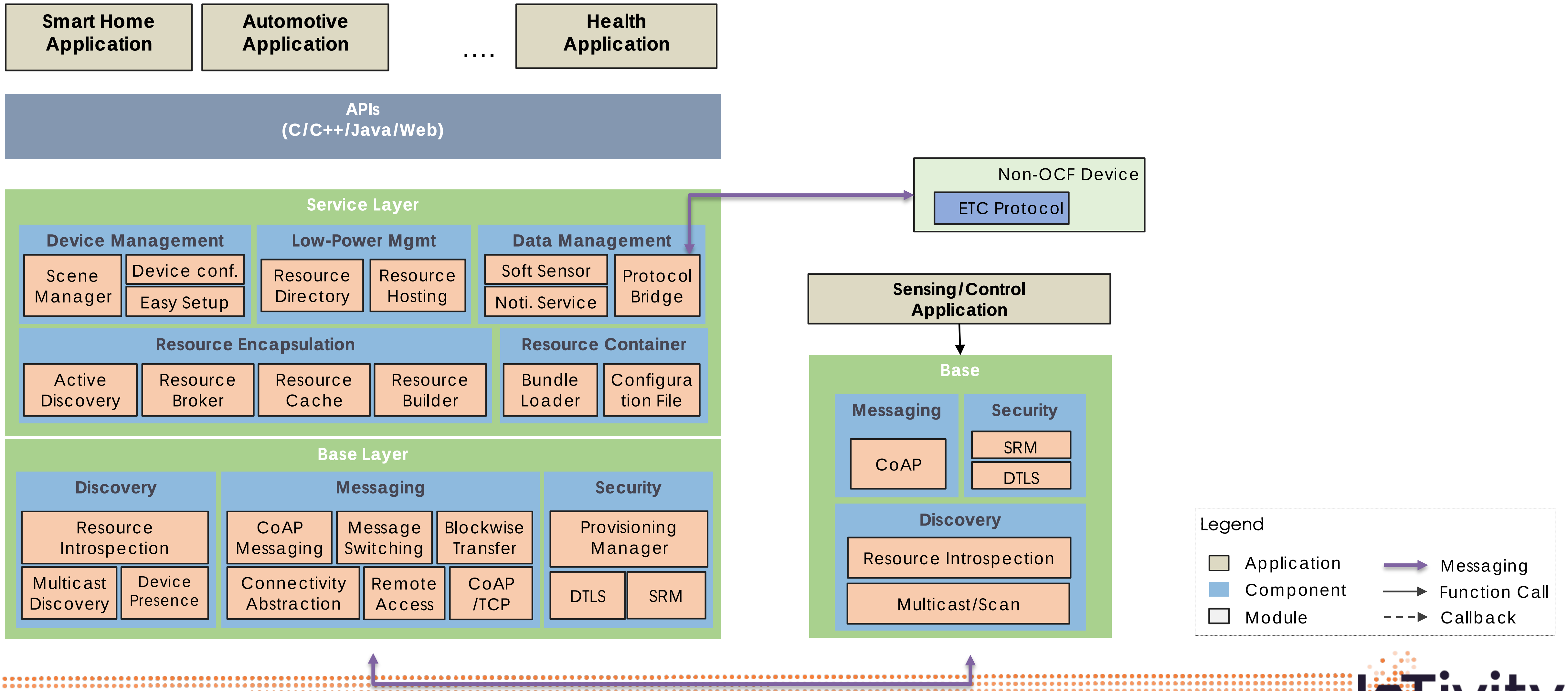
IoTivity 2.1

- Cloud to Cloud Interface
- Pub-Sub
- DDS Messaging Support

Appendix

OPEN
CONNECTIVITY
FOUNDATION™

IoTivity – Deeper View



Smart Home Application

Automotive Application

....

Health Application

APIs
(C/C++/Java/Web)

Service Layer

- Device Management: Scene Manager, Device conf., Easy Setup
- Low-Power Mgmt: Resource Directory, Resource Hosting
- Data Management: Soft Sensor, Noti. Service, Protocol Bridge
- Resource Encapsulation: Active Discovery, Resource Broker, Resource Cache, Resource Builder
- Resource Container: Bundle Loader, Configuration File

Base Layer

- Discovery: Resource Introspection, Multicast Discovery, Device Presence
- Messaging: CoAP Messaging, Message Switching, Blockwise Transfer, Connectivity Abstraction, Remote Access, CoAP /TCP
- Security: Provisioning Manager, DTLS, SRM

Non-OCF Device

- ETC Protocol

Sensing/Control Application

Base

- Messaging: CoAP
- Security: SRM, DTLS
- Discovery: Resource Introspection, Multicast/Scan

Legend

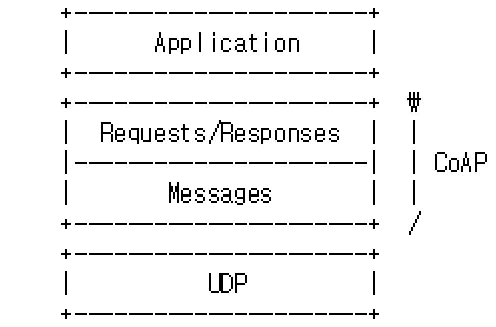
□	Application	⟶	Messaging
🟦	Component	⟶	Function Call
□	Module	- - -▸	Callback

IoTivity

Messaging - CoAP Messaging

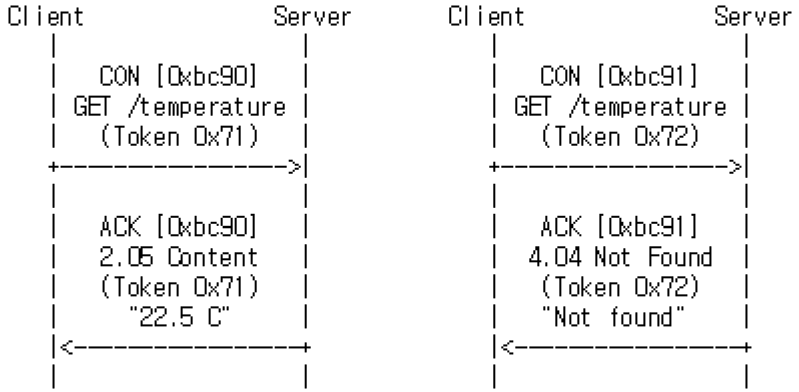
OPEN CONNECTIVITY FOUNDATION™

Message Architecture



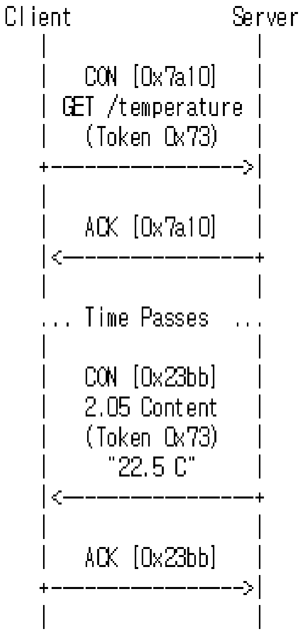
```
+----------------------+
|      Application     |
+----------------------+
+----------------------+  \
|  Requests/Responses  |  |
|----------------------|  | CoAP
|       Messages       |  |
+----------------------+  /
+----------------------+
|          UDP         |
+----------------------+
```

Figure 1: Abstract Layering of CoAP



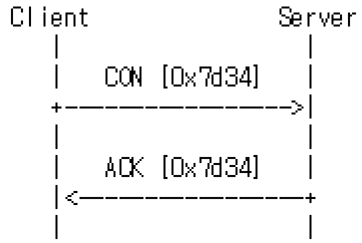
```
Client              Server       Client              Server
   |                  |             |                  |
   |   CON [0xbc90]   |             |   CON [0xbc91]   |
   | GET /temperature |             | GET /temperature |
   |   (Token 0x71)   |             |   (Token 0x72)   |
   +----------------->|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

Figure 4: Two GET Requests with Piggybacked Responses



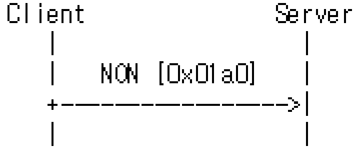
```
Client              Server
   |                  |
   |   CON [0x7a10]   |
   | GET /temperature |
   |   (Token 0x73)   |
   +----------------->|
   |                  |
   |   ACK [0x7a10]   |
   |<-----------------+
   |                  |
   ... Time Passes  ...
   |                  |
   | 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

Figure 5: A GET Request with a Separate Response



```
Client              Server
   |                  |
   |   CON [0x7d34]   |
   +----------------->|
   |                  |
   |   ACK [0x7d34]   |
   |<-----------------+
   |                  |
```

Figure 2: Reliable Message Transmission



```
Client              Server
   |                  |
   |   NON [0x01a0]   |
   +----------------->|
   |                  |
```

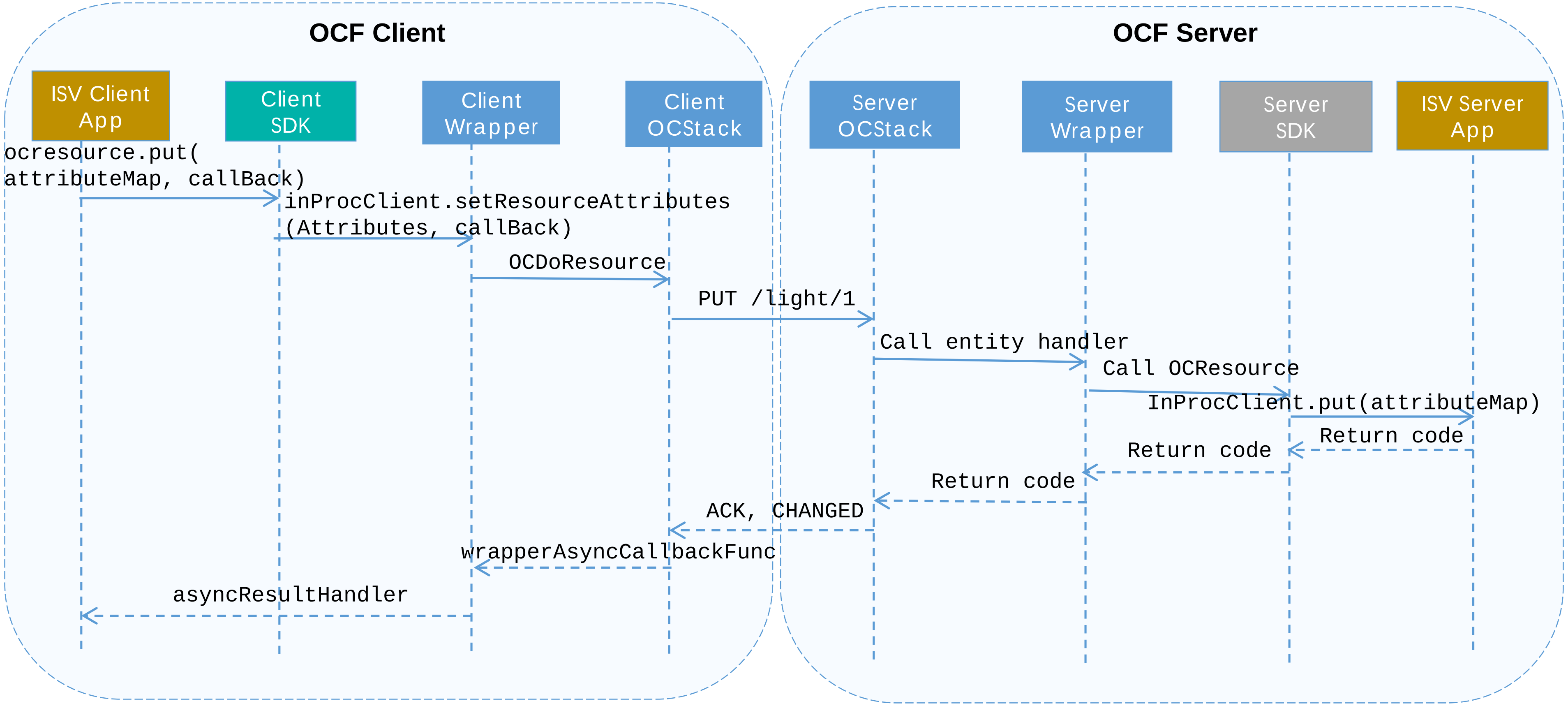
Figure 3: Unreliable Message Transmission

- ❖ IETF Standard, RFC 7252, Constrained Application Protocol
- ❖ Web transfer protocol for use with constrained nodes & constrained network.
- ❖ Designed for M2M scenarios
- ❖ Request/response (piggyback style) interaction between application endpoint

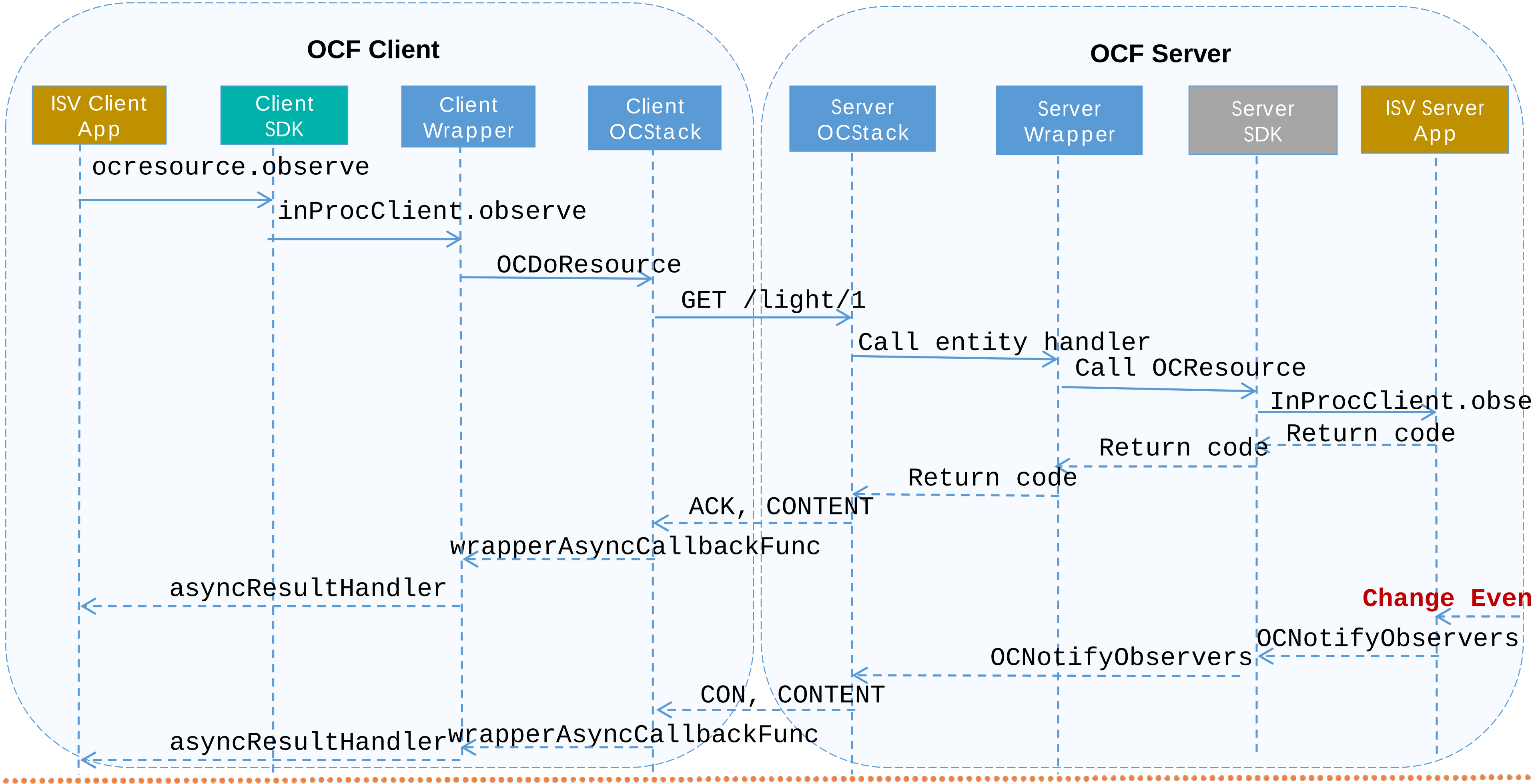
Description (Reference: https://tools.ietf.org/html/rfc7252)

IoTivity

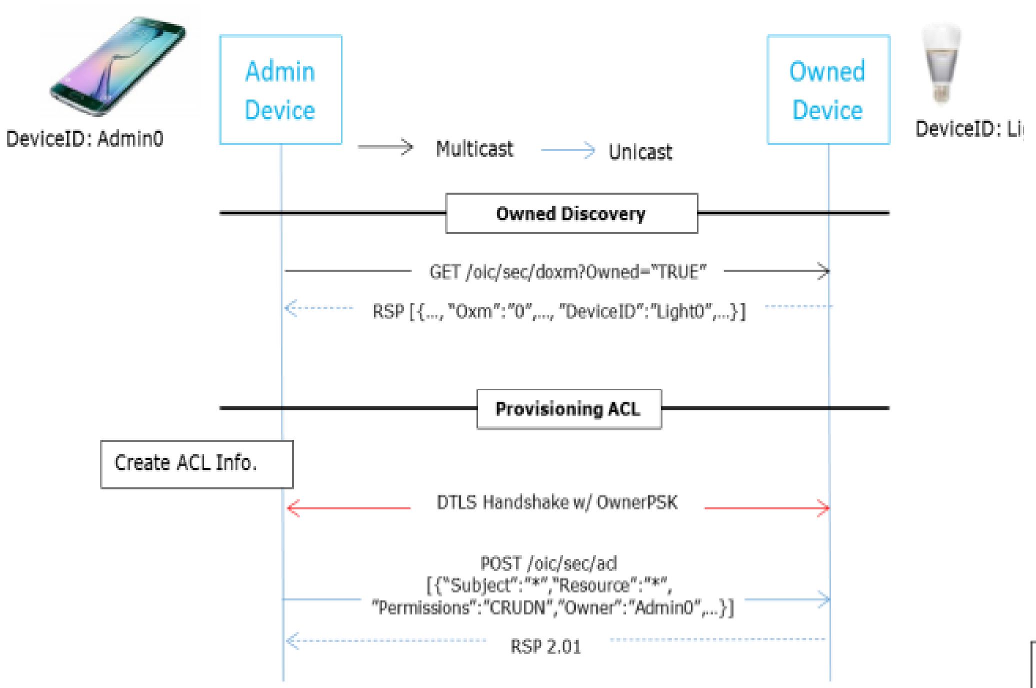
Setting a Resource State – Sequence Diagram



Observing Resource State



Onboarding & Provisioning Call Flow



DeviceID: Admin0

Admin Device

Owned Device

DeviceID: Li

→ Multicast

→ Unicast

Owned Discovery

GET /oic/sec/doxm?Owned="TRUE"

RSP [{...,"Oxm":"0",..., "DeviceID":"Light0",...}]

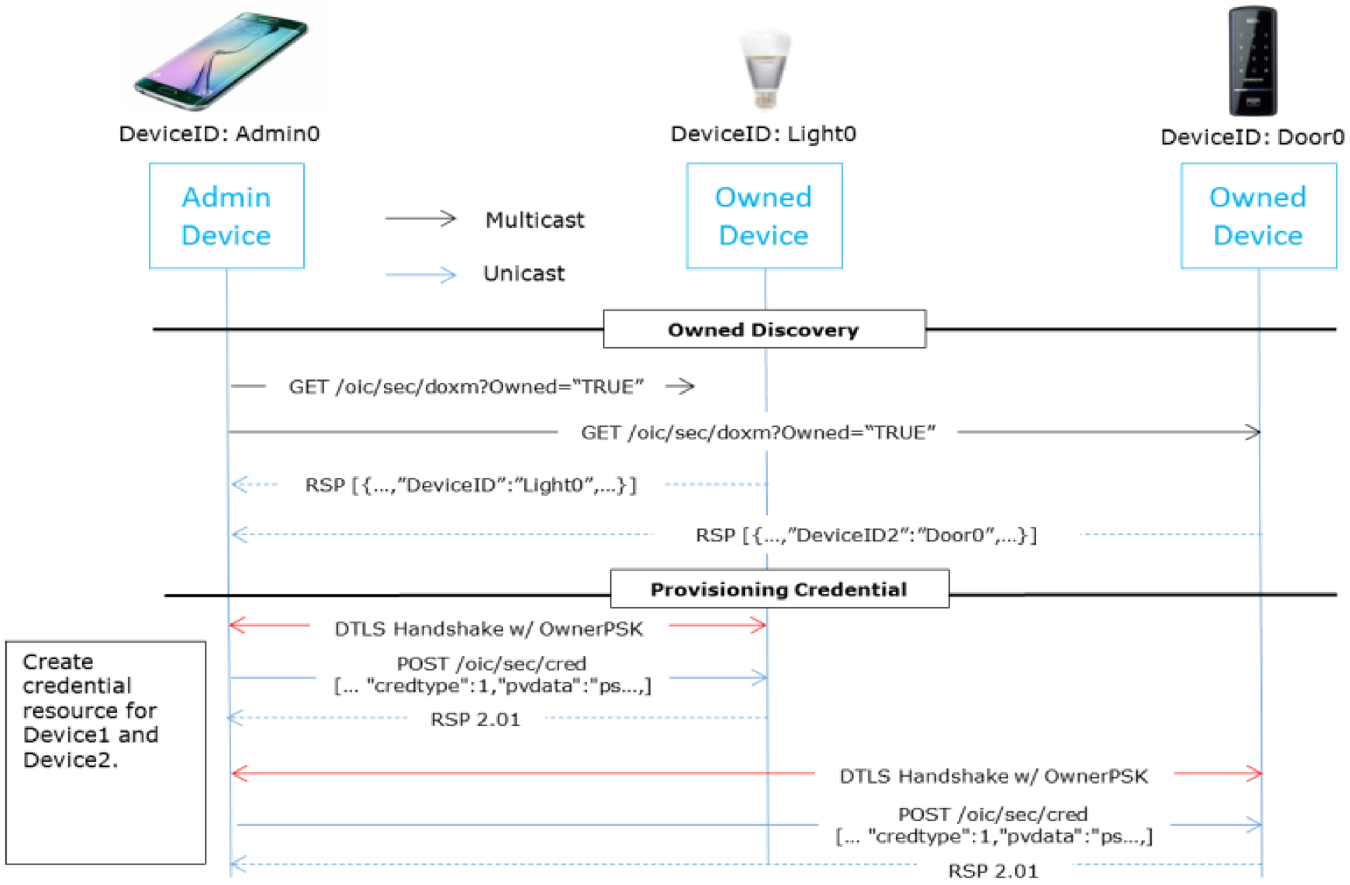
Provisioning ACL

Create ACL Info.

DTLS Handshake w/ OwnerPSK

POST /oic/sec/acl
[{"Subject":"*","Resource":"*",
"Permissions":"CRUDN","Owner":"Admin0",...}]

RSP 2.01



DeviceID: Admin0

DeviceID: Light0

DeviceID: Door0

Admin Device

Owned Device

Owned Device

→ Multicast

→ Unicast

Owned Discovery

GET /oic/sec/doxm?Owned="TRUE"

GET /oic/sec/doxm?Owned="TRUE"

RSP [{...,"DeviceID":"Light0",...}]

RSP [{...,"DeviceID2":"Door0",...}]

Provisioning Credential

Create credential resource for Device1 and Device2.

DTLS Handshake w/ OwnerPSK

POST /oic/sec/cred
[... "credtype":1,"pvdata":"ps...,]

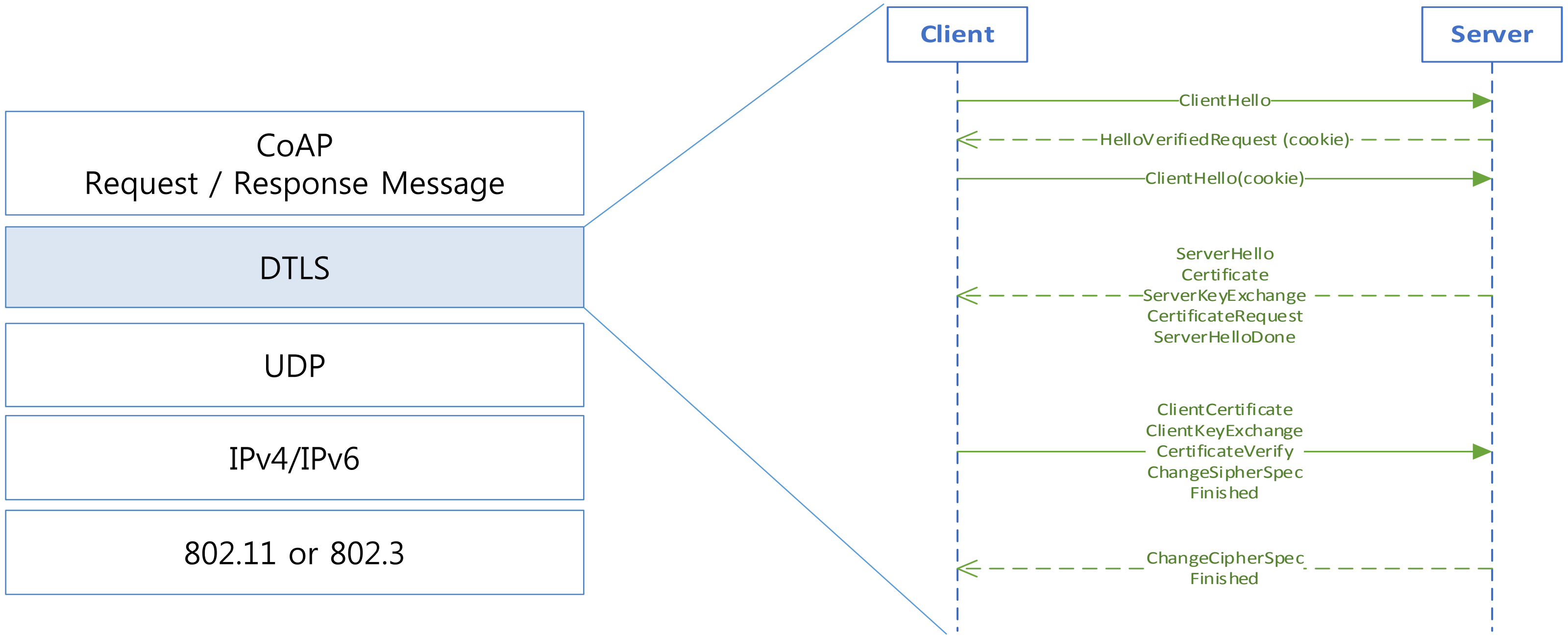
RSP 2.01

DTLS Handshake w/ OwnerPSK

POST /oic/sec/cred
[... "credtype":1,"pvdata":"ps...,]

RSP 2.01

Secure Communication



CoAP
Request / Response Message

DTLS

UDP

IPv4/IPv6

802.11 or 802.3

Client

Server

ClientHello

HelloVerifiedRequest (cookie)

ClientHello(cookie)

ServerHello
Certificate
ServerKeyExchange
CertificateRequest
ServerHelloDone

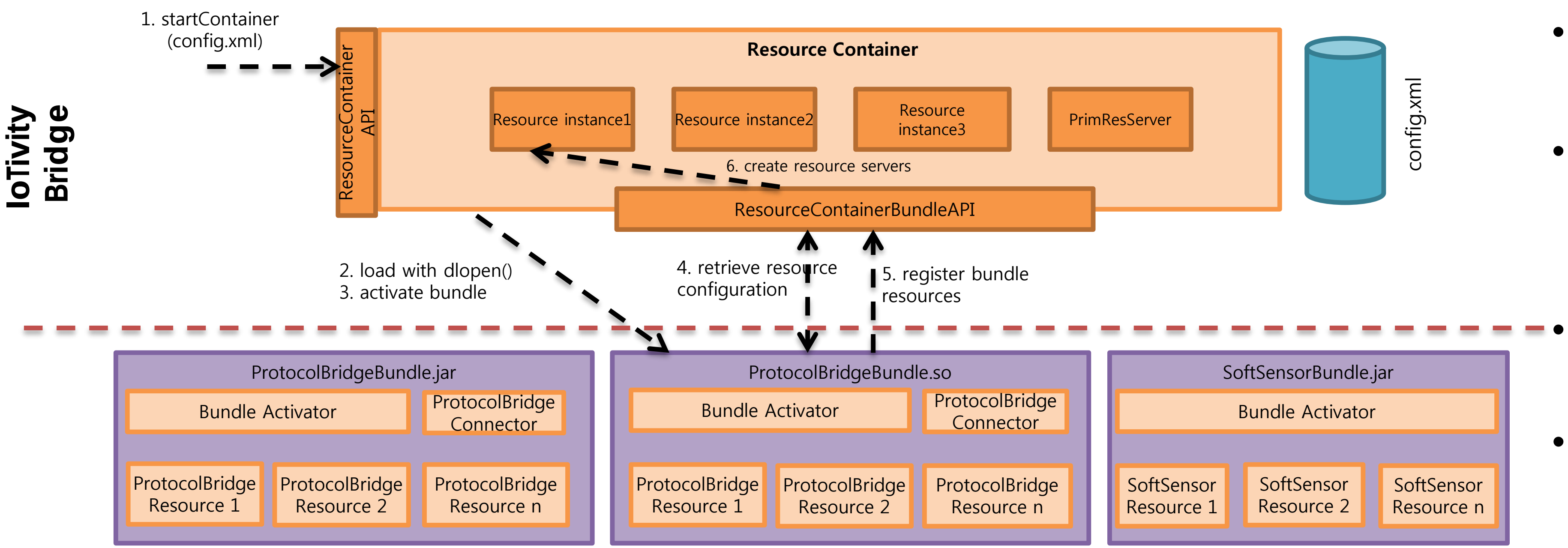
ClientCertificate
ClientKeyExchange
CertificateVerify
ChangeSipherSpec
Finished

ChangeCipherSpec
Finished

Cipher Suites & Mechanism Supported

- **Authentication**: Pre-Shared keys (PSK) or Certificate
- **Message Confidentiality & Integrity**: TLS_PSK_AES_128_CCM_8
- **Replay protection**: MAC includes sequence number
- **Scalability** : tiny-DTLS for Constraint Device

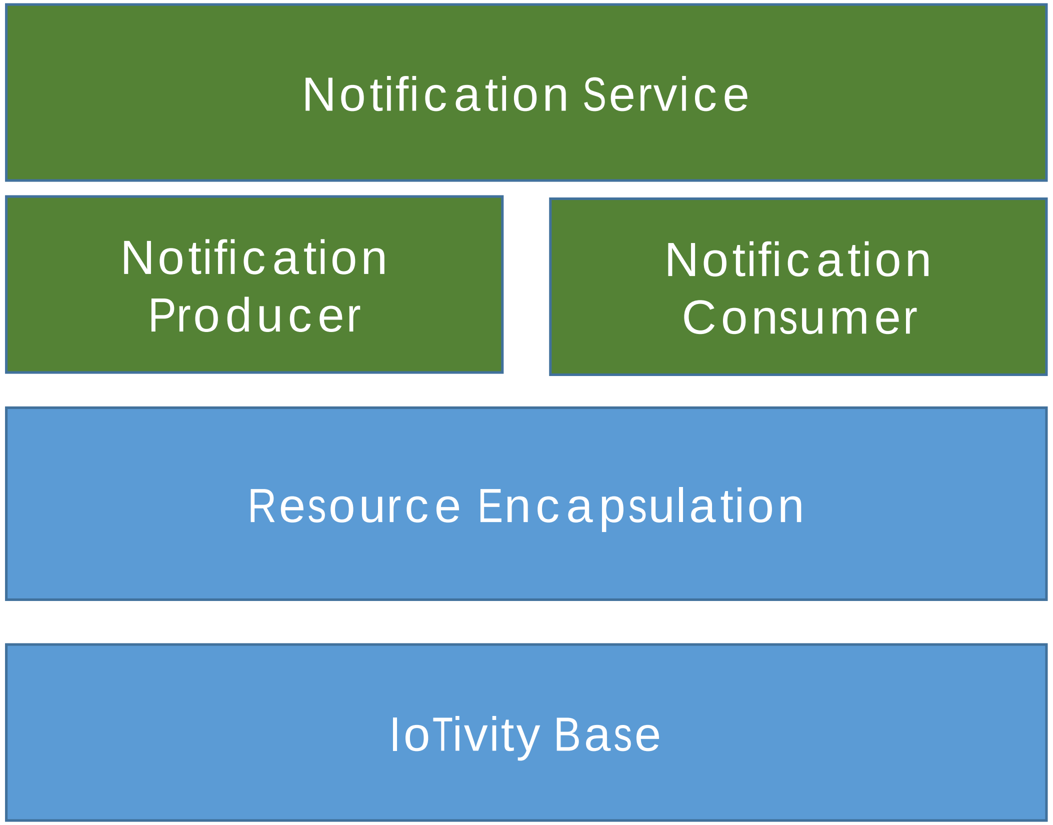
Resource Container



- Integrates non-OCF resources (Bundle)
- Handles dynamic loading of resource bundles & dynamic creation of resources
- Supports C++ .so files & Java .jar files
- Common configuration for bundles and configured resources

Notification Service

OPEN
CONNECTIVITY
FOUNDATION™



- Rich Notification Delivery (Text, Audio, Video)
- Uniform Notification Information across platforms (Linux, Android, Tizen)
- Notification Delivery acknowledgement from consumer to producer

IoTivity