IoTivity Architecture

Ashok Subash
Samsung Electronics R & D Institute Bangalore
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

OCF Client → OCF Server
P2P Direct

OCF Client
XMPP/STUN/TURN/ICE
OCF Intermediary
Gateway
OCF Servers
Remote Access

Cloud
CoAP over TCP
OCF Servers
Cloud based intelligent Services
IoTivity – High Level Architecture

**Key Goals**

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

**APIs**

(C/C++/Java/JS)

**Service Layer**

- Device Management
- Low-Power Management
- Data Management
- Resource Encapsulation
- Resource Container

**Base Layer**

- Discovery
- Messaging
- Security
- Discovery

**Lite Device**

Sensing/Control Application

**Rich Device**

Consumer | Enterprise | Industrial | Automotive | Health

**APIs**

(C/C++/Java/JS)

**Key Goals**

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

**IoTivity Profiles**

**IoTivity Framework**

**IoTivity Connectivities**

**Open Connectivity Foundation™**

**IoTivity**

- Bluetooth
- Wi-Fi
- IEEE 802.15.4
- ZigBee
- Z-Wave
- ANT+
- Remote Access
- Cloud
IoTivity Base Layer & APIs
Discovery Subsystem

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

- **OCF Server**
  - announce resource “OCF/server”
  - multicast listen [port 5683]
- **OCF Client**
  - multicast listen [port 5683]

[Figure 2] Multicast/Unicast over WiFi / Ethernet

- **OCF Server**
  - multicast listen [port 5683]
  - unicast response “OCF/server”
  - advertise OCF service
- **OCF Client**
  - multicast listen [port 5683]
  - find resource
  - scan OCF service

[Figure 3] Advertise/Scan over BLE/BT

- **OCF Server**
  - advertise OCF service
  - response “OCF/server”
- **OCF Client**
  - scan OCF service
  - find resource

**Connectivity** | **Discovery Mechanism** | **Description**
--- | --- | ---
WiFi & Ethernet (over IP) | IP Multicast | CoAP Multicast Port: 5683 (Assigned by IANA)
 & CoAP Secure Port: 5684
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

**Discovery within local network**

---

OCF: Open Connectivity Foundation
IANA: Internet Assigned Numbers Authority
Discovery – Finding a Resource

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);

// Devices that matches the query answers as indicated below

Sequence Diagram

OCF Client

Light 192.168.1.11

GET /oc/core?rt=light
(IP multicast)

Light 192.168.1.12

GET /oc/core?rt=light
(multicast)

Fan 192.168.1.21

GET /oc/core?rt=light
(multicast)

ACK, CONTENT

ACK, CONTENT

GET /oc/core?rt=light
(multicast)
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

Legend
- CA Component
- CA Module
- External
- Ubuntu
- Android
- Tizen
- Arduino
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

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

GW/Proxy

S1 (LW)  S2 (LW)  S3 (cRA)

Local subnet

STUN/TURN  Signaling Server  DM Server

RA Server

C1 (RA)
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 NATboxes
  - 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
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

*Destination-Seqenced Distance-Vector Routing
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

Querying a Resource State: Sequence Diagram
IoTivity Security
Key Functionality

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

Security Subsystem Architecture

- Resource Introspection (RI) layer
- Secure Resource Manager (SRM) layer
- Provisioning Manager (PM)
  - Ownership Transfer Manager (OTM)
  - Provisioning Database Manager (PDM)
- Secure Resource Provider (SRP)
- Connectivity Abstraction (CA) layer
- DTLS modules, etc.

- Secure Virtual Database
- Provisioning Manager (Admin Device)
- Persistent Storage Interface (PSI)
- Resource Manager (RM)
- Policy Engine (PE)
- Secure Resource Provider (SRP)
- Resource Access over DTLS

- Access Denied
- Access Approved

*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

Diagram:
- Applications
- Primitive Services
- IoTivity Base

Diagram shows various components including:
- Multi-Phy Easy Setup
- Soft Sensor Manager
- Protocol Bridge
- Notification Service
- Scene Manager
- Resource Directory
- Resource Container
- Resource Server Builder
- Resource Broker
## Resource Encapsulation

### Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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

1. Loads configuration
   - containerConfig.xml (resource instance specific configuration)

2. Loads resource bundles
   - hueToOCF.so

- 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 with 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

\[\text{e.g. Away Home – All Lights turned off, Doors locked} \]
\[\text{Watching Movie – Living Room lights off, TV On, Speaker On} \]

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

Scene Manager

Resource Encapsulation

IoTivity Base

"Away Home" Scene

LED Bulb

Thermostat

Door Lock

Scene List

URI:/oic/SceneList

Scene Collection

URI:/oic/SceneCollection/1

Scene Member

URI:/oic/SceneMember/1

Scene Member

URI:/oic/SceneMember/2

Resource Model

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

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

Thin Device enhances its lifetime delegating its resource subscriber to richer hosting device
Low Power Management – Resource Directory

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

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

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

Device 3 - Constrained device that needs to sleep and cannot respond to multicast discovery queries

Multicast query request
MultiPhy Easy Setup

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

**Scenario**

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

**Network type** (onboarding)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Network Type</th>
<th>SSID</th>
<th>Passphrase type</th>
<th>Passphrase</th>
<th>Target network type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WiFi STA</td>
<td>OIC-network</td>
<td>WPA-PSK2</td>
<td>password</td>
<td>WiFi STA</td>
</tr>
<tr>
<td>2</td>
<td>WiFi SoRiP</td>
<td>OIC-network</td>
<td>WPA-PSK2</td>
<td>password</td>
<td>WiFi STA</td>
</tr>
</tbody>
</table>

**Create SoRiP**

Connected

**QR Code**

Read QR Code

---

**IoTivity**

22
Simulator Service

Simulating different OCF resources

- OCF resources can be simulated, using resource model definition (RAML) files.
- Manages creation, deletion, request handling and notifications for 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
IoTivity 1.1.0
- Scene Manager
- Direct Pairing
- Support for NFC
- IoTivity Cloud Support

IoTivity 2.0
- CoAP-HTTP Proxy
- Integration with Thread connectivity
- Notification Service

IoTivity 2.1
- Cloud to Cloud Interface
- Pub-Sub
- DDS Messaging Support

March 2016

Sep 2016

Not finalized
IoTivity – Deeper View
Messaging - CoAP Messaging

**Message Architecture**

- **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**

Setting a Resource State – Sequence Diagram

ocresource.put(attributeMap, callBack)

OCF Client

Client SDK
Client Wrapper
Client OCStack

ocresource.put(attributeMap, callBack)

Client OCStack
Client Wrapper
Client SDK

ocresource.put(attributeMap, callBack)

OCF Server

Server OCStack
Server Wrapper
Server SDK

call entity handler

ISV Client App
ISV Server App

OCF Client

inProcClient.setResourceAttributes(Attributes, callBack)

OCF Server

PUT /light/1

Return code

ACK, CHANGED

wrapperAsyncCallbackFunc

asyncResultHandler

Return code

Return code

Return code

Return code
Observing Resource State

- **OCF Client**
  - ISV Client App
  - Client SDK
  - Client Wrapper
  - Client OCStack
  - `ocresource.observe`
  - `inProcClient.observe`
  - `OCDoResource`
  - `GET /light/1`
  - `asyncResultHandler`

- **OCF Server**
  - Server OCStack
  - Server Wrapper
  - Server SDK
  - ISV Server App
  - Call entity handler
  - Call `OCResource`
  - `InProcClient.observe`
  - `OCNotifyObservers`

- **Change Event**
  - `ACK, CONTENT`
  - `wrapperAsyncCallbackFunc`
  - `Return code`
  - `CON, CONTENT`
  - `asyncResultHandler`
Onboarding & Provisioning Call Flow
Secure Communication

- **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

1. startContainer (config.xml)
2. load with dlopen()
3. activate bundle
4. retrieve resource configuration
5. register bundle resources
6. create resource servers
Notification Service

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