IoTivity – Connecting Things with IoT

Ashok Subash
Samsung Electronics
Agenda

- Overview of IoT, OIC & IoTivity
- IoT Stack, Topologies & Protocol
- IoTivity Architecture
- Programming IoTivity Core APIs
- Programming IoTivity Service APIs
- Summary
The Internet of Things is the network of physical objects that contains embedded technology to communicate and sense or interact with the objects' internal state or the external environment.*

*Gartner, July 2014
What makes IoT Possible?

• H/W Miniaturization & Lower BOM Cost
• Advancements in Sensor Technology
• Low Power Connectivity Technologies
• IP as key Interoperability Protocol
• Devices ability to run on battery for longer duration (> 10 years)
What’s hindering IoT?

- Non IP based standards technology resulting in limited Interoperability
- Proprietary Protocols & Technologies
- Licensing issues
- Companies creating “Closed Ecosystem” (Zero or limited Interoperability with other Vendor devices)
- Low adoption of Open Standards by various Industry Consortiums
- Security & Privacy concerns
- Dilemma in “Ownership” of Data from variety of IoT devices
- Lack of Strong Certification for ensuring Protocol & Application Profile Interoperability

Need Standards & Reference Implementation which cater to these issues
OIC & IoTivity

- **Open Interconnect Consortium (OIC)**
  - Founded by Leading Technology Companies including Cisco, GE, Intel, MediaTek & Samsung
  - Defines standards for connectivity requirements
  - Ensures interoperability of billions of Internet of Things (IoT) devices.

- **IoTivity**
  - An open source software framework implementing OIC Standards
  - Ensures seamless device-to-device connectivity to address the emerging needs of the Internet of Things.
Key Focus of OIC

- OIC Standards addresses multiple vertical domains including Home Automation, Automotive, Enterprise, HealthCare, Industrial scenarios

- Initial focus on Smart Home & Office solutions

- Adopt Open Standards like IETF when applicable & standardize on areas, not addressed

http://openinterconnect.org/
Key Focus of IoTivity

- Open Source Framework implementing OIC Standards
- Licensed under Apache License Version 2.0
- Available on TIZEN, Android, Arduino, Linux(Ubuntu) Platforms
- Provide APIs at 2 Levels
  - IoTivity Base
  - IoTivity Services

https://www.iotivity.org
TIZEN based IoT EcoSystem

IoT Cloud & Analytics

IoT Communication Protocols & Framework

OS Platform

TIZEN Multi Platform Profile

H/W

ARTIK 1

ARTIK 5

ARTIK 10
IoT Stack, Topologies & Protocol
**IoT Technology Stack – End to End - Executive View**

**Identity & Security**
Tools that manage user authentication and system access, as well as secure the product, connectivity, and product cloud layers.

**PRODUCT CLOUD**
- **Smart Product Applications**
  - Rules/Analytics Engine
  - Application Platform
  - Product Data Database

**CONNECTIVITY**
- **Network Communication**
The Protocols that enable communication between the product and the cloud.

**PRODUCT**
- **Product Software**
  - An embedded operating System, onboard software applications, an enhanced user interface and product control components

- **Product Hardware**
  - Embedded Sensors, processors and a connectivity port/antenna that supplement traditional mechanical and electrical components

**External Information Sources**
- A Gateway for information from external sources such as weather, traffic, commodity and energy prices, social media and geo mapping – that informs the product capabilities.

**Integration with Business Systems**
- Tools that integrate data from smart connected products with core enterprise business systems such as ERP, CRM and PLM.

---

*Harvard Business Review – Nov 2014*
IoT Technology Stack – Technical View

IoT Profiles
- Consumer
- Enterprise
- Industrial
- Automotive
- Health

IoT Framework
- Discovery
- Data Transmission
- Device Management
- Data Management
- Security

IoT Connectivity
- Bluetooth
- Wi-Fi
- IEEE 802.15.4
- ZigBee
- WAVE
- ANT+
- Remote Access
- Cloud

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

- P2P Direct
  - OIC Client
  - OIC Server

- OIC Client
  - XMPP/STUN/TURN/ICE
  - OIC Intermediary
  - Gateway
  - OIC Servers
  - Remote Access

- OIC Client
  - Cloud
  - Gateway
  - Cloud based intelligent Services
  - OIC Servers

Smart Device
Smart Appliance
Sensors, Wearables
Concept of Resource & RESTful Architecture

- Client-Server
- Stateless
- Cache
- Uniform Interface
- Layered System

Resource: Any information that can be named
e.g. Document, Image, a collection of other resources, non-virtual objects (Things)
Constrained Application Protocol (CoAP)

- Open IETF Standard (RFC 7252)
- Compact 4 Byte Header
- UDP (Default), SMS, TCP Support
- Strong DTLS Security
- Asynchronous Subscription
- Built-In Discovery
Constrained Environments – What is it?

- Limitations on Code Size (ROM/Flash)
- Size of State & Buffers (RAM)
- Processing Power (CPU)
- Power Consumption
- User Interface & Accessibility in deployment

Constrained Nodes

- Low achievable bitrate/throughput
- High packet loss & high variability of packet loss (delivery rate)
- Highly asymmetric link characteristics
- Severe penalties for using larger packets
- Limits on reachability over time
- Lack of advanced services such as IP multicast

Constrained Networks
IoTivity Architecture
IoTivity – High Level Architecture

Application Profiles
(Smart Home, Smart Health, Smart Retail, Auto)

OIC Services
- Things Manager
- Resource Container
- Notification Manager

OIC Core
- Resource Introspection
- Messaging
- Discovery
- Security
- Connectivity Abstraction

Service

Connectivity Abstraction

Smart Devices (OIC Clients)
(Smartphone, SmartTV, SmartHub etc)

Thin Device (OIC Server)
(Thermostat, Motion Sensor etc)
# IoTivity Framework – Key Functionality

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>IoTivity discovery supports multiple discovery mechanisms for devices and resources in proximity and remotely</td>
</tr>
<tr>
<td>Data Transmission</td>
<td>IoTivity data transmission supports information exchange and control based on a messaging and streaming model</td>
</tr>
<tr>
<td>Data Management</td>
<td>IoTivity data management supports the collection, storage and analysis of data from various resources.</td>
</tr>
<tr>
<td>Device Management</td>
<td>IoTivity device management supports configuration, provisioning and diagnostics of devices.</td>
</tr>
</tbody>
</table>
**IoTivity Module View**

1. **Application**
   - Service SDK (C++, Java)
   - Things Manager
   - IoTivity Base (C++ SDK)

2. **IoTivity Base (C SDK)**
   - Resource Introspection
   - Secure Resource Manager
   - Connectivity Abstraction
     - CoAP (libcoap)
     - tinydtls
   - Thin Block (Constrained Devices)
     - IoTivity Base (C SDK)
     - Resource Introspection
     - Secure Resource Manager
     - Connectivity Abstraction
       - CoAP (libcoap)
       - tinydtls
     - UDP/IP
     - BT
     - BLE
Programming IoTivity Core APIs
Steps involved in using IoTivity Core API

- Registering a Resource
- Finding a Resource
- Querying a Resource State
- Setting a Resource State
- Observing Resource State
Registering a Resource – Call Flow

```
OCPlatform::registerResource(resourceHandle, "/light/1", "core.light", Oic.if.baseline", entityHandlerCb, OC_DISCOVERABLE | OC_OBSERVABLE);
```

```
OCCreateResource(resourceHandle, resourceTypeName.c_str(), resourceInterface.c_str(), resourceURI.c_str(), EntityHandlerWrapper, NULL, resourceProperties);
```
Registering a Resource – Sequence Diagram

ISV Server App

SDK

ocplatform.registerResource

InProcServer.registerResource

Server Wrapper

OCStack

Success/Failure

OCCreateResource

OCStackResult

Success/Failure

Success/Failure
Finding a Resource – Call Flow

1. Application

2. C++ API (SDK)
   - OCPlatform::findResource(host, "/light/1", connectivityType, resourceHandlerCb);
   - OCDoResource(resourceHandle, OC_REST_GET, "/light/1", 0, payLoad, connectivityType, qos, &cbData, headerOptions, numOptions);
   - CASendRequest(endPoint, &requestInfo);

3. Multicast

4. Sends a multicast query

5. //Devices that matches the query answers as indicated below
Finding a Resource – System Sequence Diagram

Client

Light 192.168.1.11
GET /oc/core?rt=light (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

ACK, CONTENT
Querying Resource State [GET]

OIC Client

- ISV Client App
- IoTivity SDK
- Client Wrapper
- Client OCStack
- OCResource.get(callBack)
- OCDoResource
- GET /light/1
- Call entity handler
- Call OCResource
- Return code
- Return code

OIC Server

- Server OCStack
- Server Wrapper
- IoTivity SDK
- ISV Server App
- wrapperAsyncCallbackFund
- wrapperAsyncCallbackFund
- wrapperAsyncCallbackFund
- asyncResultHandler
- ACK, CONTENT
- Return code
Setting a Resource State – Sequence Diagram

Client SDK

Client Wrapper

Client OCStack

ocresource.put(attributeMap, callBack)

inProcClient.setResourceAttributes(Attributes, callBack)

OCDoResource

PUT /light/1

Return code

ISV Client App

OIC Client

ISV Server App

OIC Server

OCStack

Server Wrapper

Server SDK

OCStack

Call entity handler

Call OCResource

InProcClient.put(attributeMap)

Return code

ACK, CHANGED

wrapperAsyncCallbackFunc

asyncResultHandler

inProcClient.setResourceAttributes(Attributes, callBack)

OCDoResource

PUT /light/1

Return code

inProcClient.setResourceAttributes(Attributes, callBack)

OCDoResource

PUT /light/1

Return code

ocresource.put(attributeMap, callBack)

inProcClient.setResourceAttributes(Attributes, callBack)
Observing Resource State

**OIC Client**
- ISV Client App
  - ocresource.observe
  - inProcClient.observe
- Client SDK
- Client Wrapper
- Client OCStack
- ocDoResource
- GET /light/1
- Call entity handler
  - Call OCResource
- wrapperAsyncCallbackFunc
- asyncResultHandler

**OIC Server**
- ISV Server App
- Server SDK
- Server Wrapper
- Server OCStack
- OCNotifyObservers
- Return code
- Change Event
- ocresource.observe
- inProcClient.observe
- ocDoResource
- GET /light/1
- Call entity handler
  - Call OCResource
- wrapperAsyncCallbackFunc
- asyncResultHandler
- asyncResultHandler

**ISV App**
- OIC Client
- OIC Server
Programming IoTivity Service APIs
IoTivity Core & Primitive Services Update

**Application**
- Service SDK (C++, Java)
  - Things Manager
  - Notification Manager

**Resource Model**
- Resource Introspection
- Secure Resource Manager
- CoAP (libcoap)
- Blockwise Transfer
- Multihop Routing
- tinydtls

**IoTivity Base (C++ SDK)**
- Resource Container
- Resource Broker
- Resource Cache
- Server Builder
- Resource Client Wrapper

**IoTivity Base (C SDK)**
- Connectivity Abstraction

**UDP/IP**
- BT
- BLE
Things Manager

- **Group Management**
  - Find candidate devices to form a group
  - Create a group of found devices
  - Create a group action for the group
  - Execute the group action

### Action Set

- **Application**
  - createGroup
  - findCandidateResources
  - OnFindResourceCb
  - joinGroup
  - Return code

- **Things Manager**
  - registerResource
  - findResource

- **IoTivity Base**
  - returncode
  - createGroup
  - registerResource
  - OnFindResourceCb
  - joinGroup
  - Return code

### URI

- OutofHome
  - uri=coap://10.251.44.228:49858/a/light|power=on*
  - uri=coap://10.252.44.221:49458/b/aircon|power=off

### Collection

- **Resource Type**
  - Thermostat
  - Air conditioner
  - LED Bulb

### Return Code

- findCandidateResources
- findResource
- OnFindResourceCb
- joinGroup
- bindResource
- Return code
Things Manager – Configuration & Diagnostics

• **Things Configuration**
  ✓ Server Side - Bootstrapping requisite information from a bootstrap server to access other IoT services
  ✓ Client Side – Getting/Updating system configuration parameters from/to multiple remote things

• **Things Diagnostics**
  ✓ Factory reset to restore all configuration parameters to default one
  ✓ Reboot to request a system rebooting
Notification Manager

- Service on Unconstrained device host resources for other Lite/Thin devices
- Hosting device mirrors resources from other Lite devices by Observing the presence & changes in other sources
Notification Manager – System Sequence Diagram

1. Register Resource
2. Start Hosting
3. On Presence
4. Find Resource
5. On Found Resource
6. Register Virtual Resource
7. Observe Hosted Resource
8. UnRegister Observe
9. UnRegister Observe

Lite Device

Hosting Device
Resource Encapsulation

- Integrates non-OIC protocols as resources
- Handles dynamic loading of resource bundles & dynamic creation of resources
- Remote resource discovery & presence check
- Provides easier way to create resource Servers
- Monitors value of attributes & automatically transmits a message
- Provide consistent reachability mgmt for discovered resource of interest
- Maintains last information of remote resource
- Data centric APIs (send/receive message → getter/setter, data cache)
Resource Server Builder

App Logic

1. Initialize Resource
2. Set Properties
3. Register Resource

Resource Server Builder

4. Register Resource as Discoverable
5. Request for Resource (get/put/post/delete/observe)
6. Handle Request
7. Send Response
8. Send Response

IoTivity Base

- Provide “data centric” API for users to set/create attributes of a resource server
- Notification for observers
- Developer does not need to deal with low-level details of CoAP communication
- Resources are defined based on properties & developer has to provide getter/setter methods
1. Discover/Monitor Resource

2. Discover Resource

3. Found Resource

4. Check Resource Presence periodically

5. Send Request

6. Process Response

7. Process Response

8. Notify Presence Change (only if presence status changes)

- Provides presence check for resource of interest

- Consistent reachability management for the resources of interest (resource duplication detection and rediscover when temporally unreachable)
Resource Cache

- Stores and updates latest resource data from remote resource
- Provides “data centric” interfaces (changes from messaging API to data getter-like interface) to resource client users

App Logic

1. Request to Cache Resource
6. Notify Cache Status

Resource Broker

2. Observe request to Resource
5. Cache Resource Data
7. Send Frequent Resource Request
10. Process Response (Update Cache)

IoTivity Base

3. Send Request
4. Resource Data
8. Send Request
9. Process Response
Resource Container

- Feature
  - Integrates non-OIC resources into the OIC ecosystem
  - Handles dynamic loading of resource bundles and dynamic creation of resources
  - Supports C++ .so files and Java .jar files
  - Common configuration for bundles and configured resources
Resource Container Interaction Flow

1. Server Builder parses the configuration file.
2. The resource container is initialized.
3. The protocol bridge is activated.
4. Resources are configured and resources are registered.
5. The container initialization is completed.
6. For each configured resource, a bundle resource is created and resources are registered.
7. The register resource process is completed.
8. The process of creating bundle resources is completed.

System integrator/user

Device Manufacturer (Bridge)

Application (e.g. oic_bridge.exe)

Resource Container

Protocol Bridge Connector

Bundle Provider (e.g. Philips Hue)

Resource Bundle.so (e.g. Philips Hue)
OIC & IoTivity – Road Ahead

IoTivity 0.9.0 (Dec 2014)
- Device & Resource Discovery IPv4
- OIC Resource Model
- ID/Addressing
- Messaging (CoAP)
- Payload encoding JSON

IoTivity 0.9.2 (August 2015)
- Initial IPv6 Support
- BLE Support
- Initial support for Remote Access
- Client-driven credential & ACL Provisioning
- Subject-based Access Control
- Resource Encapsulation
- Payload encoding CBOR

IoTivity 1.0 (October 2015)
- Multi-Phy Easy Setup
- Cloud Data Interface (CoAP over TCP)
- Resource Directory
- Simulator
- Security Updates
- Block wise Transfer
- Multi-Hop Routing Manager
## OIC & IoTivity – Road Ahead

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Phy Easy Setup</td>
<td>• Connect Out-of-box device without UI onto network &amp; provisioning</td>
</tr>
<tr>
<td>Data Interface to Cloud</td>
<td>• Actuation of devices from Cloud Apps, Collection of Sensor Data in Cloud</td>
</tr>
</tbody>
</table>
| Resource Directory    | • Constrained device that needs to sleep and cannot respond to multicast discovery queries  
                          • Power constrained device that cannot keep responding to multicast queries                                                             |
| Simulator             | • Developers can test implementations without having real hardware  
                          • Manufactures can provide reference profiles using simulator  
                          • Enables users/developers to test the functionality of the device/profile first before purchasing the real device  
                          • Easy for manufacturers to test the profiles with the test suite before releasing the profiles. |
## OIC & IoTivity – Road Ahead

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
</table>
| Security Updates              | • Filter Resource requests  
• Access control of resources  
• Secure Transmission of data across variety of IoT devices  
• Certificate based Key Mgmt |
| Blockwise Transfer            | • Send/Receive of Larger data over IoTivity Stack                           |
| Multi Hop Routing Manager     | • Routing of packets across variety of connectivities                      |
| AV Streaming                  | • Audio Video Streaming                                                    |
| IPv6 & 6LowPAN support        | • Supporting IPv6 and 6LoWPAN as part of IoTivity Connectivity Abstraction  |
Summary

- IoT Landscape, Roles of OIC & IoTivity
- Understanding the big picture in IoT including various topologies
- Architectural Principles & Key Protocols adopted by OIC & IoTivity
- High Level Architecture of IoTivity Stack & types of Deployment
- Programming using IoTivity Base APIs
- Programming using IoTivity Service APIs
- Ongoing & Future work
Thank you