IoTivity – Connecting Things in IoT

Ashok Subash
Agenda

- Overview of IoT, OIC & IoTivity
- IoT Stack, Topologies & Protocol
- IoTivity Architecture
- Programming IoTivity Core APIs
- Programming IoTivity Service APIs
- Demo
Internet of Things – What is it?

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.*

- Vehicle, Asset, Person, Pet Monitoring & Controlling
- Smart Agriculture
- Energy Consumption
- Security & Surveillance
- Building Management
- M2M & Wireless Sensor Network
- Wearables & Things
- Smart Home & Cities
- Telemedicine & Healthcare

*Gartner, July 2014
What is making 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 (> 5 years)
What is 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
TIZEN IoT Ecosystem

- IoT Cloud & Analytics
- IoT Communication Protocols & Framework
  - IoTivity
- 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+S
- 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
OIC Servers
Cloud based
intelligent Services

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)

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
- Soft Sensor Manager
- Notification Manager
- Protocol Plugin Mgr

OIC Core
- Resource Introspection
- Messaging
- Discovery
- Security

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>
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

<table>
<thead>
<tr>
<th>Application</th>
<th>C++ API (SDK)</th>
<th>C API (SDK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OCPlatform::registerResource(resourceHandle, &quot;/light/1&quot;, &quot;core.light&quot;, &quot;oic.if.baseline&quot;, entityHandlerCb, OC_DISCOVERABLE</td>
<td>OCCreateResource(resourceHandle, resourceTypeName.c_str(), resourceInterface.c_str(), resourceURI.c_str(), EntityHandlerWrapper, NULL, resourceProperties);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JSON/GBOR Encode/Decoder, OCStack, CoAP, Connectivity Manager</td>
</tr>
</tbody>
</table>
Registering a Resource – Sequence Diagram

ISV Server App

ocplatform.registerResource

SDK

InProcServer.registerResource

Success/Failure

Server Wrapper

Success/Failure

OCStack

OCCreateResource

OCStackResult
Finding a Resource – Call Flow

1. Application
2. C++ API (SDK)
3. JSON/CBOR Encode/Decoder
4. OCStack
5. CoAP
6. Connectivity Manager

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
Finding a Resource – System Sequence Diagram

Client

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

Light 192.168.1.11

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

ACK, CONTENT

Light 192.168.1.12

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

Fan 192.168.1.21

ACK, CONTENT
Finding a Resource – Sequence Diagram

**ISV Server App**
- `ocplatform.findResource`

**SDK**
- `client.ListenForResource`
- `asyncCallbackFunc`

**Client Wrapper**
- `OCDoResource`
- `wrapperAsyncCallbackFunc`

**OCStack**
- Request via multicast
  - Unicast Response From 192.168.1.11
- Unicast Response From 192.168.1.12

**CoAP**
- `wrapperAsyncCallbackFunc`
Querying Resource State [GET]

ISV Client App

IoTivity SDK

Client Wrapper

Client OCStack

OIC Client

ocresource.get(callBack)

clientWrapper.get(callBack)

OCDoResource

GET /light/1

Call entity handler

Call OCResource

Call OCResource

Return code

Return code

Return code

ISV Server App

OIC Server

Server OCStack

Server Wrapper

IoTivity SDK

ResultHandler

wrapperAsyncCallbackFun

ACK, CONTENT

asyncResultHandler

OIC Client

OIC Server
Setting a Resource State – Sequence Diagram

OIC Client

- ISV Client App
- Client SDK
- Client Wrapper
- Client OCStack

OIC Server

- Server OCStack
- Server Wrapper
- Server SDK
- ISV Server App

ocresource.put(attributeMap, callBack)

inProcClient.setResourceAttributes(Attributes, callBack)

OCDoResource

PUT /light/1

Call entity handler

Call OCResource

Return code

Return code

ACK, CHANGED

Return code

wrapperAsyncCallbackFunc

asyncResultHandler
Observing Resource State

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

**OIC Server**
- Server OCStack
- Server Wrapper
- Server SDK
- ISV Server App
- OCNotifyObservers
- InProcClient.observe
- Return code
- Call entity handler
- Call OCResource
- Return code
- CON, CONTENT
- ocresource.observe
- wrapperAsyncCallbackFunc

**Change Event**
Programming IoTivity Service APIs
IoTivity Services

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

- OIC Services
  - Things Manager
  - Soft Sensor Manager
  - Notification Manager
  - Protocol Plugin Manager

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

- Connectivity Abstraction

- Protocol Plugin Manager
  - Communication with non-IoTivity devices

- Notification Manager
  - Provides resource hosting function

- Soft Sensor Manager
  - Sensor Data Collection, Aggregation & Fusion

- Things Manager
  - Group Creation & Management

- IoTivity Service Description

<table>
<thead>
<tr>
<th>IoTivity Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Things Manager</td>
<td>Group Creation &amp; Management</td>
</tr>
<tr>
<td>Soft Sensor Manager</td>
<td>Sensor Data Collection, Aggregation &amp; Fusion</td>
</tr>
<tr>
<td>Notification Manager</td>
<td>Provides resource hosting function</td>
</tr>
<tr>
<td>Protocol Plugin Manager</td>
<td>Communication with non-IoTivity devices</td>
</tr>
</tbody>
</table>
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

```
LED Bulb
25 C
Thermostat
Air conditioner
```

Action Set
- **Name**: OutofHome
  - *uri=coap://10.251.44.228:49858/a/light|power=on*
  - *uri=coap://10.252.44.221:49458/b/aircon|power=off*
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
Soft Sensor Manager

- Helps in observing changes to Physical Sensors
- Allows developers to create Soft Sensor
- Utilizes data from multiple physical sensors to make sense of data from end user perspective. E.g. DiscomfortIndex

Smart Device
  - Application
  - Soft Sensor Manager
  - Iotivity Base

Thin Device
  - Iotivity Base
  - Physical Sensor App

Aggregation & Sensor Data Fusion

Calculate “Discomfort Index”

Raw Sensor Data
Soft Sensor Manager – SDK Class Diagram

- IDReader
- IMModelData

<<interface>>
IQueryEngineEvent

virtual onQueryEngineEvent (int cqid, IDataReader *pResult)

<<callback to listener>>

Application
onQueryEngineEvent

SSMInterface

IQueryEngineEvent*
mQueryEngineEventListener

startSSMCore
registerQuery
unregisterQuery
stopSSMCore
Protocol Plugin Manager

- Allows OIC compliant Applications to communicate with non OIC device protocols
- Uses Plugin Manager APIs to find, start, stop plugins
- Uses IoTivity APIs to find and operate on resources similar to interaction with OIC device
Protocol Plugin Manager – Overall Flow

Application Process

Plugin A

Protocol A

Plug A

Protocol B

Plugin B

IoTivity Base API

OIC Protocol

Load/Unloads Plugins dynamically

findResource

Application Logic

startPlugin("resource type","oic.fan"

Plugin Manager API (C/C++/Java APIs)

Plugin Manager

Plugin Abstraction

Pluff

OSGi (Felix)
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

2. Stop Hosting

9. UnRegister Observe
# OIC & IoTivity – Road Ahead

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
</table>
| IoTivity Security             | • Filter Resource requests  
                                | • Access control of resources  
                                | • Secure Transmission of data across variety of IoT devices                                                                 |
| Device Onboarding & Provisioning | • Connect Out-of-box device without UI onto network & provisioning                                                                             |
| IPv6 & 6LowPAN support        | • Supporting IPv6 and 6LoWPAN as part of IoTivity Connectivity Abstraction                                                                 |
| AV Streaming                  | • Audio Video Streaming                                                                                                                        |
| Blockwise Transfer            | • Send/Receive of Larger data over IoTivity Stack                                                                                             |
| Routing through Heterogeneous transports | • Routing of packets across variety of connectivities                                                                                    |
| Data Interface to Cloud       | • Actuation of devices from Cloud Apps, Collection of Sensor Data in Cloud                                                                     |
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
Usecase Description

- As Bob’s car approaches home, proximity sensor sense his presence & garage door opens automatically
- As Bob approaches main door, it gets unlocked after exchanging appropriate credentials
- Meanwhile Bob’s Z1 phone connects with home gateway and turns on group of devices like light and fan based on his preset preference
- Based on increase in temperature, the fan in room is started

- Same scenario is executed with another group of devices for another user, with their preferences
Use Case Pictorially!

Garage with Proximity Sensor

Arriving home

Group A

Group B

IoTivity

IoTivity

1

2
Demo Setup

- Proximity Enabled Door
- Sandwich maker
- Light
- Fan
- Gateway

IoTivity Smart Device
IoTivity Thin Device
Demo
Thank you