Tizen/Artik IoT Lecture Chapter 13. IoTivity Cloud

Sungkyunkwan University

Contents

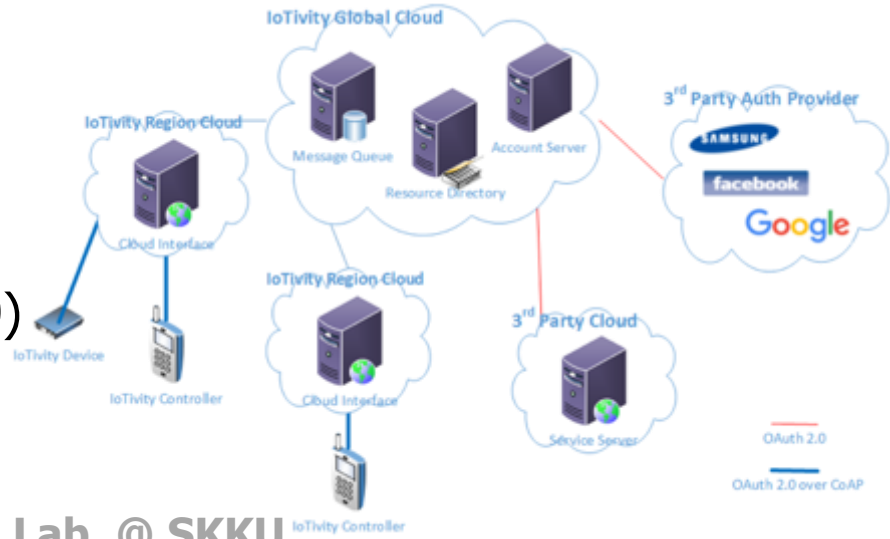
- **IoTivity Cloud**
  - Architecture
  - Features
- **IoTivity Cloud SW Stack**
- **IoTivity Cloud in Resource Model**
- **Source Tree**
- **IoTivity Cloud API**
- **Air Conditioner Sample**

IoTivity Cloud

- **Extending accessibility of IoTivity devices over local network (with authentication)**
- **Senarios**
  - **Easy-Setup**: configuration of the network of unboxed thing devices without input methods
  - **Remote Control**: communication between resource server and client over network region
    - **Event Notification** from OIC device to cloud server
    - **Device control command** from cloud server to OIC device
  - **Service Integration(TBD)**: allowing 3rd party service provider to see and control the resource server

IoTivity Cloud Architecture

- **Resource Server/Client**
  - IoTivity-enabled devices
  - Handle session through CoAP over TCP/TLS to CI server
- **Cloud Interface(CI) Server (Region Cloud)**
  - Region-based server
  - Accept connection from clients
  - Receive notification data
- **Global Cloud**
  - Cluster region clouds
  - Provide authentication (OAuth 2.0)
  - Registration of resources



IoTivity Cloud Features

- **OAuth 2.0 over CoAP**
  - Authentication for resource registration & access
- **CoAP over TCP**
- **App-level KeepAlive**
- **Cloud-centric Interfaces**
  - Resource registration, discovery, update, delete, presence
- **Reqeust Queue Broker**
  - in order to support PUB/SUB
- **Netty Framework (used in Server Base Layer)**
  - Asynchronous event-driven framework
  - Support various network protocols for easy socket programming

Features: OAuth 2.0 over CoAP

- **OAuth 2.0 (IETF RFC6749)**
  - Authorization framework to allow **3rd party app to access to HTTP service restrictively**
- **OAuth 2.0 over CoAP**
  - OAuth 2.0 framework based on **CoAP**, not HTTP

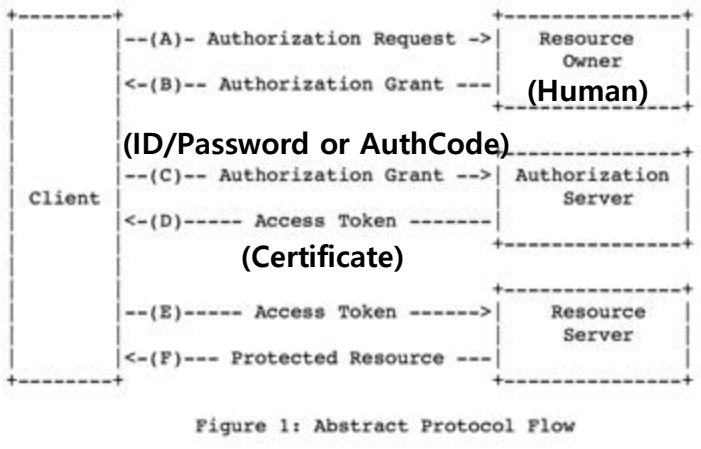
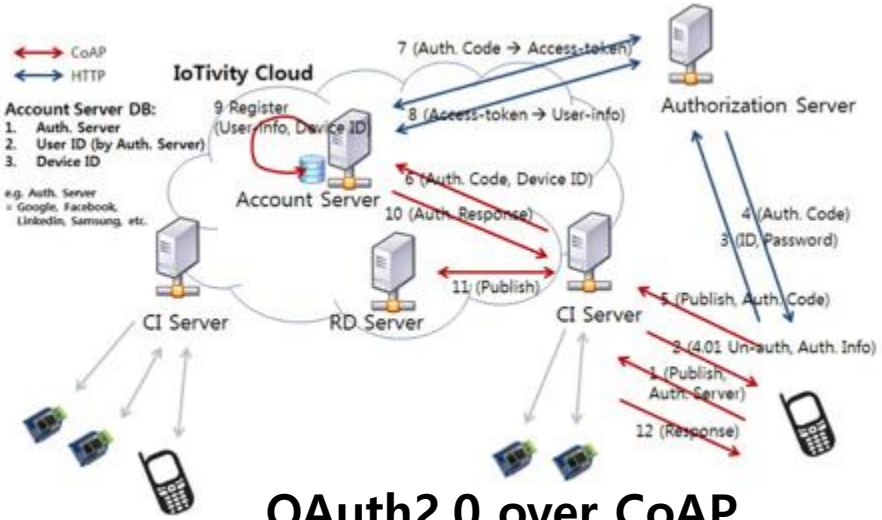


Figure 1: Abstract Protocol Flow

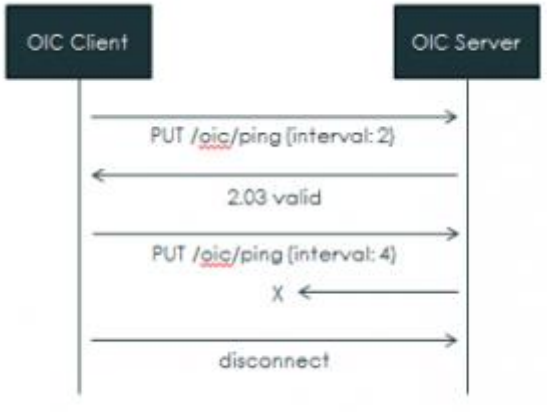
Original OAuth2



OAuth2.0 over CoAP
(Example of Device Registration)

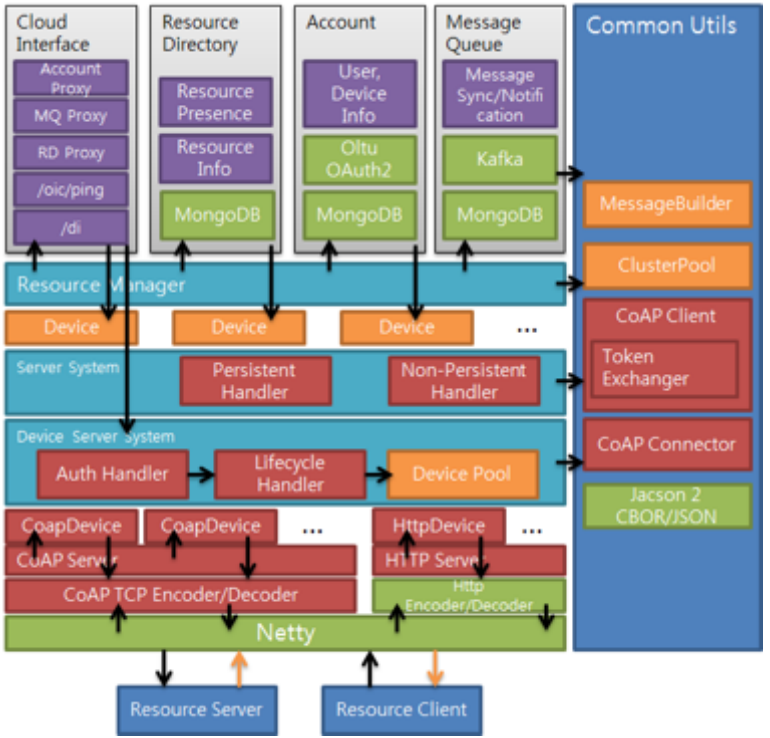
Other Features

- **CoAP over TCP (#)**
  - Originally, CoAP is designed to run on UDP.
  - UDP is NOT appropriate for Cloud-scale network
    - Reliable Delivery, Congestion Control, Flow Control Mechanism
- **App-level KeepAlive**
  - **KeepAlive**: Recognize disconnection of network session by interaction of simple messages
  - **Motivation**: Limits of TCP-level KeepAlive
    - No consideration on app's lifecycle
      - ex. After app's crash, stop to send KeepAlive message
    - Configuration on KeepAlive is kernel-dependent
      - ex. Interval of sending KeepAlive message



IoTivity Cloud SW Stack

- **Server Base Stack**
  - Based on Netty framework
  - CoAP over TCP encoder/decoder
- **Cloud Interface Server**
  - Server-side OAuth 2.0 handshake
  - KeepAlive resource server
- **Resource Directory Server**
  - Provides resource registration, discovery, update, delete to CI server
  - Resource information DB
- **Account Server**
  - Manages User & Access Token
- **Message Queue Broker**
  - PUB/SUB interaction



IoTivity Cloud in Resource Model

- **KeepAlive Resource**

Resource Name	URI	Resource Type
Keep-Alive	/oic/ping	oic.wk.ping

Property Name	Value Type	Access Modes	Description
in	integer	W	The time interval for which connection shall be kept alive and not closed e.g. {in : 1}
inarray	Array integers	R	Array of time intervals from Server to Device e.g. {inarray : [1,2,4,8]}

- **Authentication Resource**
  - Will be Specified in OIC Spec v1.1
- **Resource Directory Resource**
  - Will be Specified in OIC Spec v1.1

IoTivity Cloud: Source Tree

- **cloud (Global/region cloud stack, client sample)**
  - **account**: Account Server process (Java)
  - **certificate**: Certificate files for IoTivity Cloud
  - **interface**: Cloud Interface server process (Java)
  - **messagequeue**: Message Queue server process (Java)
  - **resourcedirectory**: Resource Directory server process (Java)
  - **samples**: **air conditioner sample**, thin light sample **(C++)**
  - **stack**: common stack for all server instances (Java)
- **resource (RI Layer)**
  - **include, src**: OCAccountManager

IoTivity Cloud API 1/2

- **Common API in Thing/User**
  - **constructAccountManagerObject()**
    - After connecting to Account Manager server, make object to point server
  - **AccountManager::signUp()**
    - Acquire access right on service (AccessToken)
    - Input: AuthProvider addr, AuthCode → Output: *AccessToken*
    - AuthCode
      - A string made by Auth Service when login with Auth Service ID/PW
      - Auth Service mediates to other 3rd party services(IoTivity Cloud Service)
  - **AccountManager::signIn()**
    - Acquire AccessToken then connect to the service
    - Input: User ID, *AccessToken* → Output: None

IoTivity Cloud API 2/2

- **Thing-side API**
  - publishResourceToRD()
    - Register local resource to the cloud server
- **User-side API**
  - Resource Introspection API
  - findResource(), post(), get(), put(), ...

Air Conditioner Sample: Controllee

1. Sign-up & sign-in to cloud's Account Server

cloud/samples/client/airconditioner/aircon_controllee.cpp

```
533     string host = "coap+tcp://";
534     host += argv[1];
535
536     OCAccountManager::Ptr accountMgr = OCPlatform::constructAccountManagerObject(host,
537                                        CT_ADAPTER_TCP);
538
539     mutex blocker;
540     unique_lock<std::mutex> lock(blocker);
541
542     if (argc == 5)
543     {
544         accountMgr->signIn(argv[2], argv[3], &handleLoginoutCB);
545         g_callbackLock.wait(lock);
546     }
547     else
548     {
549         accountMgr->signUp(argv[2], argv[3], &handleLoginoutCB);
550         g_callbackLock.wait(lock);
551         accountMgr->signIn(g_uid, g_accesstoken, &handleLoginoutCB);
552         g_callbackLock.wait(lock);
553     }
```

If controllee has *already acquired* Access Token, just do **sign-in**.

If it has *NOT acquired* Access Token, do **sign-up** to acquire Access Token then do **sign-in**.

Air Conditioner Sample: Controllee

2. Define resources representing air conditioner

```
558     AirConditionerResource  airConditioner("/sec/aircon/0", { "x.com.samsung.da.device" }, { DEFAULT_INT
    ERFACE, BATCH_INTERFACE, LINK_INTERFACE });
559
560     BinarySwitchResource    binarySwitch("/power/0", { "oic.r.switch.binary" }, { DEFAULT_INTERFACE });
561
562     TemperatureResource     temperature("/temperature/0", { "oic.r.temperature" }, { DEFAULT_INTERFACE }
    );
563
```

3. Register resource to local OCStack

```
568     result = OCPlatform::registerResource(airConditioner.m_handle,
569                                           uri,
570                                           rt,
571                                           itf,
572                                           std::bind(&AirConditionerResource::entityHandler
573                                                   , &airConditioner, std::placeholders::_1),
574                                           OC_DISCOVERABLE);
```

```
624     result = airConditioner.addChildResource(&binarySwitch);
625
626     result = airConditioner.addChildResource(&temperature);
```

Air Conditioner Sample: Controllee

3. Register Device Info Resource

```
631     ResourceHandles resourceHandles;
632
633     OCDeviceInfo        devInfoAirConditioner;
634     OCStringLL          deviceType;
635
636     deviceType.value = "oic.d.airconditioner";
637     deviceType.next = NULL;
638     devInfoAirConditioner.deviceName = "FAC_2016";
639     devInfoAirConditioner.types = &deviceType;
640     devInfoAirConditioner.specVersion = NULL;
641     devInfoAirConditioner.dataModelVersions = NULL;
642
643     OCPlatform::registerDeviceInfo(devInfoAirConditioner);
```

4. Publish the resources to cloud's Resource Directory

```
645     result = OCPlatform::publishResourceToRD(host, OCConnectivityType::CT_ADAPTER_TCP,
646                 resourceHandles,
647                 &onPublish);
```

```
653     result = OCPlatform::publishResourceToRD(host, OCConnectivityType::CT_ADAPTER_TCP,
654                 resourceHandles,
655                 &onPublish);
```

Air Conditioner Sample: Controllee

5. Turn on/off air conditioner.

– When turning on the air conditioner, notify its state change to observers

```
666     while (true)
667     {
668         cin >> cmd;
669         OCRepresentation    rep;
670
671         switch (cmd[0])
672         {
673             case '1':
674                 rep.setValue(string("value"), true);
675                 binarySwitch.setBinarySwitchRepresentation(rep);
676                 break;
677
678             case '0':
679                 rep.setValue(string("value"), false);
680                 binarySwitch.setBinarySwitchRepresentation(rep);
681                 break;
682
683             case 'q':
684                 goto exit;
685                 break;
686         }
687     }
```

```
136     void setBinarySwitchRepresentation(OCRepresentation &rep)
137     {
138         bool value;
139         if (rep.getValue("value", value))
140         {
141             m_value = value;
142             m_representation.setValue("value", m_value);
143             cout << "\t\t\t\t" << "value: " << m_value << endl;
144
145             propagate();
146         }
147     }
 97     OCStackResult propagate()
 98     {
 99         if (m_interestedObservers.size() > 0)
100         {
101             std::shared_ptr<OCResourceResponse> resourceResponse =
102             { std::make_shared<OCResourceResponse>() };
103
104             resourceResponse->setErrorCode(200);
105             resourceResponse->setResourceRepresentation(getRepresentation(), DEFAULT_INTERFACE);
106
107             return OCPlatform::notifyListOfObservers(m_handle,
108                     m_interestedObservers,
109                     resourceResponse);
110         }
111
112         return OC_STACK_OK;
113     }
```

Air Conditioner Sample: Controller

1. Sign-up & sign-in to cloud's Account Server
2. Find all resources of OIC devices

```
320     result = OCPlatform::findResource(g_host, "/oic/res?rt=oic.wk.d",
321                                       static_cast<OCConnectivityType>(CT_ADAPTER_TCP | CT_IP_USE_V4),
322                                       &foundDevice);
```

3. Turn on/off air conditioner.

```
332     while (true)
333     {
334         cin >> cmd;
335         OCRepresentation    rep;
336
337         switch (cmd[0])
338         {
339             case '1':
340                 turnOnOffSwitch(true);
341                 break;
342
343             case '0':
344                 turnOnOffSwitch(false);
345                 break;
```

```
158 void turnOnOffSwitch(bool toTurn)
159 {
160     OCRepresentation binarySwitch;
161     binarySwitch.setValue("value", toTurn);
162
163     QueryParamsMap      query;
164     g_binaryswitchResource->post("oic.r.switch.binary", DEFAULT_INTERFACE, binary
    Switch, query, &onPut);
165 }
```

cloud/samples/client/airconditioner/aircon_controller.cpp