IoTivity

For Automotive

meta-ocf-automotive tutorial

Automotive Linux Summit

#LFALS, Tokyo, Japan <2017-05-31>

Philippe Coval
Samsung Open Source Group / SRUK
philippe.coval@osg.samsung.com
こんにちは from Philippe Coval

- Software engineer for Samsung Research
  - Open Source Group, EU team (@UK + DE + FR + CZ...)
- Commit into IoTivity, Tizen
  - Plus automotive related projects: Yocto, GENIVI, AGL
- Interest: Usages, OS/hardware support, build, community
- Ask me online for help:
Agenda

- IoT interoperability
  - Open Connectivity Foundation + IoTivity
- OCF Automotive working group
  - Demos
- Learning IoTivity by examples
- Convergence of automotive data models
“Without **trust** there's no cooperation. And without **cooperation** there's no **progress**. History stops.”

~ Rick Yancey, *The Last Star*
Motivations for Interoperability in IoT

• To break **silos** between:
  – Personal devices: Mobile, Wearable...
  – Shared devices: SmartHome, Cars (IVI, many OS)
  – Infrastructure: Buildings, Cities (traffic, pedestrians...)
  – Online services and probably more to come...

• A **common** open standard is welcome!
  – To provide abstracted features:
    • Connectivity, Security, Portability, Modularity
    • Protocol, Opensource stack, Community
Open connectivity foundation's missions

- Provide software linking the Internet Of Things
  - Focus on interoperability and seamless connectivity between devices
- Write **specification**, establish a protocol (formerly named OIC)
  - Rely on existing standards (IETF: CoAP, Cbor..)
- Sponsor Reference **implementation**: **IoTivity**
  - OpenSource (Apache 2.0 license) use existing FLOSS libs
    - Hosted by Linux Foundation (like kernel, Tizen etc)
    - Rule: No unimplemented features in specification
- **Certify** products for over 300 members (join us!)
Common technology for multiple profiles

Architectures
- RESTful
- CRUD-N

Communication
- IETF
  - (CoAP, Cbor)
- Security
  - DTLS

Semantics
- RAML

Cloud: HTTP/TCP

Mobility:
- Automotive
  - OBD-II
  - CAN
  - LIN
  - MOST
  - LTE
  - 802.11p
  - 5G*

Personal Devices:
- Wearables:
  - BLE

Smart Home:
- Zigbee

Personal Devices:
- Bluetooth

Smart Home:
- Zigbee

Cloud:
- HTTP/TCP

Samsung Open Source Group

#LFALS
IoTivity Framework for connecting devices

- **Core cross platform libraries**
  - **C API**: resource layer + connectivity abstraction (IP, BT, BLE...)
    - Link to libcoap, tinycbor (code footprint ~128-KiB), + mbedtls
  - **C++ API**: C++11 bindings to build extra services
- + High level **services** (Mostly C++)
  - Data/Device Management: Container, Hosting, Encapsulation, Scene
  - Simulator (Eclipse based), http proxy
- + Plugins: Transport, Cloud Interface, Bridging
- **Related projects**
  - IoTivity-Node: Javascript bindings
  - IoTivity-constrained: For thin devices (micro-controllers)
Security matters for IoT

- Link layer provides **secure channel** (using DTLS) to connectivity abstraction (CA)
- Devices need to be owned (once or xfer) and **provisioned** using onboarding tool:
  - Establish ownership in user's network
- Secure Resource Manager (SRM):
  - Secure platform's resources
    - Device provisioning, Credentials, Access control list, Persistence
  - Policy engine: Request filtering: Grant, deny resource requests
    - Per policy, requester ID, ACL, device status...
    - Is an OIC resource (“/oic/sec/cred”)
- Hardware hardening: use encryption and secure contexts, RNG, IO etc
- Details: [https://www.slideshare.net/SamsungOSG/iot-meets-security](https://www.slideshare.net/SamsungOSG/iot-meets-security)
“Any sufficiently advanced technology is indistinguishable from magic.”

~ Arthur C. Clarke
OCF Automotive profile's mission

• Provide OCF technology for connected cars, by proposing
  – A common definition of vehicle resources
  – A common way to interact with those (inside or outside vehicle)
  – Based on or bridging to existing standards

• Cooperative effort of existing FLOSS Automotive projects
  – Tizen, GENIVI, AGL, W3C, RVI ...
SmartHome to Automotive #CES2017
https://youtu.be/3d0uZE6lHvo
"The secret of getting ahead is getting started."
~ Mark Twain
Supported Automotive OS

- Download OS image shipping IoTivity: Tizen, GDP, AGL?
  - Or install package from repository (RPM)
- Use “meta-oic” layer on OE/Yocto based distributions:
  - **GENIVI** (Specification first): integrated
  - **AGL/Automotive Grade Linux** (Code first): optional
    - Tizen Yocto project to build Common + IVI
- Rebuild package from sources for most GNU/Linux systems
  - [https://wiki.iotivity.org/build](https://wiki.iotivity.org/build)
Build Yocto's Poky with IoTivity

• Adding “meta-oic” layer to poky reference distribution

  ```
git clone http://git.yoctoproject.org/git/poky
cd poky && source ./oe-init-build-env
git clone http://git.yoctoproject.org/git/meta-oic
  ```

• Append to environment files previously generated:

  - **“poky/conf/bblayers.conf”** (Layer path file)
    ```
    RELATIVE_DIR := \\
    "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True))+’/../../../’})"
    BBLAYERS += "${RELATIVE_DIR}/meta-oic"
    ```

  - **“poky/conf/local.conf”** (Project configuration file)
    ```
    CORE_IMAGE_EXTRA_INSTALL += " packagegroup-iotivity "
    ```

• Rebuild poky image using: bitbake core-image-minimal
Check using samples apps

- Shared libs plus various validation examples are shipped:
  - find /opt/iotivity*

- Ie: playback smart light example scenario on loopback
  - Open 2 sessions (hint: use GNU screen) for client and server:
    cd /opt/iotivity/examples/resource/cpp/ && ./simpleserver
    cd /opt/iotivity/examples/resource/cpp/ && ./simpleclient

- More: https://wiki.iotivity.org/examples
Build AGL with IoTivity

- Use `repo` tool to pull sublayers (including meta-ioc):
  
  ```
  repo init -u https://gerrit.automotivelinux.org/gerrit/AGL/AGL-repo
  repo sync
  ```

- Use custom configuration script to select features (agl-full, agl-iotivity...):
  
  ```
  MACHINE=qemux86-64
  source ./meta-agl/scripts/aglsetup.sh -m $MACHINE \ agl-all-features agl-iotivity
  ```

- Use regular yocto tools:
  
  ```
  bitbake iotivity agl-demo-platform
  ROOTFS=$PWD/tmp/deploy/images/$MACHINE/agl-demo-platform-$MACHINE.ext4 \ runqemu $MACHINE; ssh ; find /opt/iotivity* # as part of OS
  ```
Build GENIVI with IoTivity

- Download distribution sources using `git`:
  ```
git clone https://github.com/GENIVI/genivi-dev-platform
  ```
- Setup using GDP custom configuration script
  ```
MACHINE=qemux86-64 # or minnowboard, raspberrypi2 etc
source ./init.sh ${MACHINE}
```
- Use regular yocto tools:
  ```
bitbake iotivity genivi-dev-platform
ROOTFS=$PWD/tmp/deploy/images/${MACHINE}/*-${MACHINE}.ext4 \
runqemu ${MACHINE} ; ssh root@$target_ip # Or use xvncviewer
find /opt/iotivity* # As part of OS
```
• Tizen is an **Operating System** based on FLOSS
  – Shipped into **consumer electronic** products
• Tizen:{3,4} part of platform (ARM/x86 32/64 .rpm)
• Tizen:2 can ship shared lib into native app (.tpk)
  – Tizen:2.4:Mobile: Samsung Z{1,2,3}
  – Tizen:2.3:Wearable: Samsung GearS{2,3}
• Tizen:3:Yocto same as poky (1.7 dizzy)
Timeline

- 2014-12-31: **meta-oic** 0.9.1 Initiated by Kishen Maloor (Intel), (with demo for edison)
- 2016-01-31: FOSDEM: Presented how to use meta-oic on Tizen Yocto (Tizen fan)
- 2016-09-14: meta-oic 1.1.1 Philippe Coval (Samsung) new contributor
- 2016-04-27: GENIVI AMM: Presented demos (fan+map+wearables on 1.1.1), +RVI
- 2016-05-08: meta-oic 1.1.1 integrated in GENIVI
- 2016-05-27: AGLF2F meeting, meta-ocf-automotive Introduced
- 2016-09-21: meta-oic 1.1.1 integrated in AGL
- 2016-12-20: meta-oic 1.2.0 integrated in GENIVI and AGL
- 2017-01-05: CES, GENIVI+Smarthome+Wearables demos (contact Sanjeev BA)
- 2017-02-04: FOSDEM: Presented “streetlight+cloud” usecase on AGL 3.0
- 2017-02-15: GENIVI announced partnership with **Open Connectivity Foundation**
- 2017-03-20: meta-oic 1.2.1+: Phil C keeps maintaining it
“Talk is cheap.
Show me the code.”
~ Linus Torvalds
What is “meta-ocf-automotive”

- Playground for OCF and Automotive **R&D experiments**
- Connecting Automotive platforms (AGL, GENIVI, Tizen)
  - Hardware: RPi {0,1,2,3}, ARTIK10, Intel, Renesas, Qualcomm…
  - to other products: SmartHome, Mobile, Wearable
- “Real world” integration/validation tests (QA)
- **Tutorial of demo codes** to learn about IoTivity, Yocto, AGL, GDP, Tizen…
  
git clone http://git.s-osg.org/meta-ocf-automotive
- More: https://wiki.iotivity.org/automotive
Prepare your environment

- Build IoTivity from sources for your OS: https://wiki.iotivity.org/os
  - Hint: build system package to use standard paths (/usr/include)
    - Or ask me for packages for Debian, Fedora, Arch (unsupported)
  - Tizen: Install OS on supported hardware and setup GBS tool
    ```
    git clone https://git.tizen.org/cgit/platform/upstream/iotivity
    cd iotivity && gbs build -P tizen_unified_armv7l -A armv7l
    ```
  - Yocto (AGL, GDP): meta-oic + meta-ocf-automotive layers to pull iotivity-example
  - Note: **Security** can be disabled at build time (for prototyping on 1.3-rel)
    - https://wiki.iotivity.org/security
iotivity-example tutorial

- OCF application developers might not develop in upstream source tree
  - SCons build system is complex (even for sample apps)
- A **standalone project** is better to get inspiration from or derivate
  - minimalist, can be used as base skeleton (fork it at will, SDK?)
- Download a **collection** of standalone subprojects:
  - `git clone http://git.s-osg.org/iotivity-example/` ; make
- Each "feature" subproject is a git module (pulling a branch based on other)
  - Nice history to understand each steps of development
  - For many OSes or build system (Currently, GNUmake, Linux, Tizen, More welcome)
“Simplicity is the ultimate sophistication.”

~ Leonardo da Vinci
Base example: Resource discovery

- branch=example/master (src/example/master/README.md)
  - Server **register** a “dummy” resource identified as “/ExampleResURI”
  - Client **discover** and list all resources' endpoints served in local network
  - GNUmake is used to build it
  - Systemd service provided to start it once installed
- branch=example/packaging is based on previous one
  - Bitbake recipe
  - RPM spec file for Tizen
  - More packaging files: Debian/Ubuntu etc
Resource discovery example flow

**IoTivity Server**

```cpp
main {
    IoTServer::init() { ModeType::Server }
    IoTServer::createResource()
        { OCPlatform::registerResource(... uri ...)
        // loop on OCProcess() is called internally
    }
}
```

**IoTivity Client(s)**

```cpp
main {
    IoTClient::init() { ModeType::Client }
    IoTClient::start()
        { OCPlatform::findResource(onFindCallback) }
    IoTClient::onFind(resource)
        { print(resource->uri) }
}
```

```
$ ./bin/server -v
(...)
log: { IoTServer::createResource(...)
log: Successfully created\   
    org.example.r.example resource
log: } OCStackResult
(...)
```

```
$ ./bin/client -v
(...)
log: { void IoTClient::onFind(...)  
log: Resource: uri: /oic/d
(...)
log: Resource: uri: /ExampleResURI
coap://[fe80::baca:3aff:fe9b:b934%25eth0]:47508
```
Geolocation example: Observation

- Branch “geolocation/master” is based on “example/packaging” and adapted:
  - Resource's URL is changed to “/GeolocationResURI”
  - **Resource type** changed to “oic.r.geolocation” (from OCF/Onelot)
  - Simulated GPS that update position continuously
- ./bin/server: is updating current position and **notifying** it
  
m_Representation.setValue();  OCPlatform::notifyAllObservers(...);
- ./bin/observer: is **observing** changes in IoTObserver::onObserve
  
  geolocation: 48.1043, -1.6715
- ./bin/client : can also get value using GET: m_OCResource->get
Derivate to Tizen app

- Port to tizen **native app**: support SDK build files, app manifest files
  - + GUI using EFL's Elementary map widget (from SDK sample)
  - Branch: “geolocation/tizen/mobile/2.4/master”
- Need to rebuild IoTivity’s **shared lib** (to be packaged) using helper:
  
  ```
  ./tizen.mk ; ls lib/*.so
  ./tizen.mk run # deploy tpk on device (ie TM1)
  ```
- More details: https://wiki.iotivity.org/tizen
Binary switch example: Boolean resource

- **Actuator**, client change value (on/off) of server's resource
  - iotivity-example's “switch/master” branch
  - based on “example/packaging” and adapted

- **Usage:**

  ```
  ./bin/server -v
  log: { OCEntityHandlerResult
  IoTServer::handleEntity(...)
  log: { OCStackResult IoTServer::handlePost(...)
  log: { void Platform::setValue(bool)
  1
  log: } void Platform::setValue(bool)
  log: { void IoTServer::postResourceRepresentation()
  (...)
  ```

  ```
  ./bin/client
  menu:
  0) Set value off
  1) Set value on
  1
  ```
Binary switch example: Resource update

```
OCPlatform::Configure(...);
OCPlatform::registerResource(...);

handleEntity(OCResourceRequest) {
    switch entityHandlerRequest->method {
        case 'POST:
            // update actuator state
            ...
    }
}
```

```
OCPlatform::post(rep, callback);
onPost(...)
```

- Client controls actuator:
  - Set resource's value
- Server is handling request
  - and responding
More examples

- **GPIO** switch to control relay attached to raspberrypi, minnowboard, ARTIK10
- **CSDK** version of binary switch
  - Arduino port (1.2-rel)
- Secured example
  - IoTivity 1.3 will have security enabled by default,
- **MRAA**: same as GPIO switch but using generic I/O Communication library
- Constrained example: (WIP) targeting MCUs (smaller than CSDK)
Constantly **talking**
isn't necessarily **communicating**

~ Charlie Kaufman
OCF Resource Model

URI:
+ common properties:
Policy
Interface...

Resource Type:
+ attribute(s)

Well knows resources URI (/oic/*):
Data models can be:

- **Described**
  - For RESTful operations (CRUD)
  - RAML+JSON, Swagger Schemas
- **Reviewed and validated**
  - OCF check consistency and versions
- **Shared**
  - [http://OneIotA.org](http://OneIotA.org) repository & tools
- **Note:**
  - IoTivity works with private models too

- **oic.r.switch.binary.json**
  - [http://www.oneiota.org/revisions/1580](http://www.oneiota.org/revisions/1580)
  - ```
  /* ... */ "definitions": {
    "oic.r.switch.binary": {
      "properties": {
        "value": {
          "type": "boolean",
          "description": "Status of the switch"
        }
      }
    }
  } */ ... */```
Aligning semantic: W3C, OCF, OMA...

• **W3C**: Automotive Working Group:
  - Vehicle Signal Specification (VSS): YAML & Json

• Many specific signals on ~100 interfaces (Chassis, OBD, Cabin, ADAS, Media...)
  - Could be dispatched over generic OCF models:
    • Switch, Speed, Distance, Movement, Audio, TimePeriod, Weight

• Example of aligned concepts:
  - oic.r.sensor.geolocation : { latitude, longitude, altitude }
    • Longitude of destination, Integer double [-180..+180]
  - oic.r.switch.binary: for lights, door, brake, belt...

• Details: [http://tinyurl.com/omaocf2017](http://tinyurl.com/omaocf2017)
OCF-VSS Translator
https://youtu.be/jKt_fPnqggo

W3C OCF Interoperability Demo
Want more ?
Smart City’s Street lights use case

- Iotivity-example (Branch “sandbox/pcoval/on/master/demo”)
  - Various examples combined in demo using nodejs
- Defective Street lights notification service:
  - Sensor reads luminance
  - Micro controller **switch** car's light on if too dark
  - **geolocation** updated continuously
  - Gateway sends message to cloud
From sensor to ARTIKCloud #FOSDEM2017
https://youtu.be/3L6_DbMLJ1k

Vehicle To Infrastructure
Proof of concept
(From devices to cloud)
https://wiki.iotivity.org/automotive

Using
Iotivity, NodeJs, ARTIK Cloud, Auto Grade Linux
CC BY SA 3.0: https://blogs.s-osg.org/author/pcoval/
A Vehicle to Infrastructure notification service

```javascript
client.on("resourcefound", function(resource) {
    if ("/GeoLocationResURI" === resource.resourcePath) {
        resource.on("update", function(resource) { gGeo = resource.properties; });
    } else if ("/IlluminanceResURI" === resource.resourcePath) {
        resource.on("update", handle);
    }
});

function handle(illuminance) {
    if (illuminance < gThreshold ) {
        var data = { illuminance: illuminance, 
                     latitude: gGeo.latitude, longitude: gGeo.longitude };
        sender.send(data); // { ARTIK's client.post(url...); }
    }
}
```

https://api.artik.cloud/
```javascript
{ 
    illuminance: 42, 
    latitude: 52.165, 
    longitude: -2.21 
}
```
Summary

- OCF establishes a **standard** for interconnecting things
  - Several profiles: SmartHome, Automotive, Health...
  - Common technology: Resource model & RESTful architecture
  - Definitions must be shared to ensure interoperability

- **Open Source** project IoTivity
  - implements it in C, C++, Java and Javascript
  - Ready to be used on Automotive Oses and beyond
  - Try using examples. Feedback welcome!
References

- Entry points:
  - http://wiki.iotivity.org/automotive
  - https://openconnectivity.org/industries/automotive
  - https://blogs.s-osg.org/tag/automotive/
  - http://git.s-osg.org/iotivity-example
  - http://git.s-osg.org/meta-ocf-automotive/

- Going further:
  - https://openconnectivity.org/resources/iotivity
  - https://openconnectivity.org/resources/oneiota-data-model-tool

- Keep in touch online:
  - https://wiki.iotivity.org/community (Wiki, Mailing list, IRC, Events ...)
  - https://wiki.tizen.org/wiki/Meeting
  - https://www.meetup.com/OCF-France/ (Local events worldwide, Soon in Tokyo)
Q&A or Extras ?
IoTivity on GENIVI demo platform:
https://youtu.be/DJKYauaOmsc
IoTivity on Automotive Grade Linux (AGL)
https://youtu.be/w_c0wxJfBsw

"IoTivity Tizen Fan" controlled by Automotive Grade Linux and TM1 on OSVehicle

https://wiki.iotivity.org/community
CC BY-SA @TizenHelper @SamsungOSG
Thank you
ありがとう
Merci !

Visit OpenConnectivity #LFALS booth !

Contact: