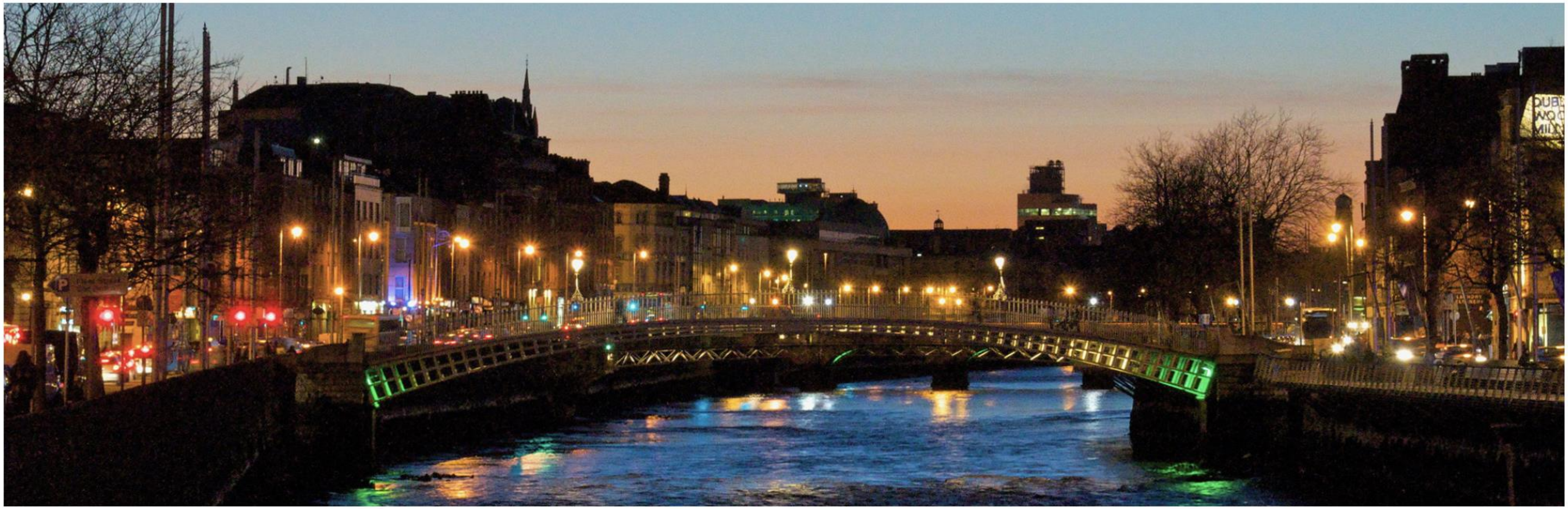
Smart Cities Architecture and Implementation Workshop: Standards and Technology



Facilitators

Kary Främling
Adjunct Professor
Aalto University,
Finland

Avleen Malhi
Postdoctoral Researcher
Aalto University,
Finland

Tuomas Kinnunen
Research Assistant
Aalto University,
Finland

Asad Javed
PhD Researcher
Aalto University,
Finland

Email: firstname.lastname@aalto.fi

Agenda

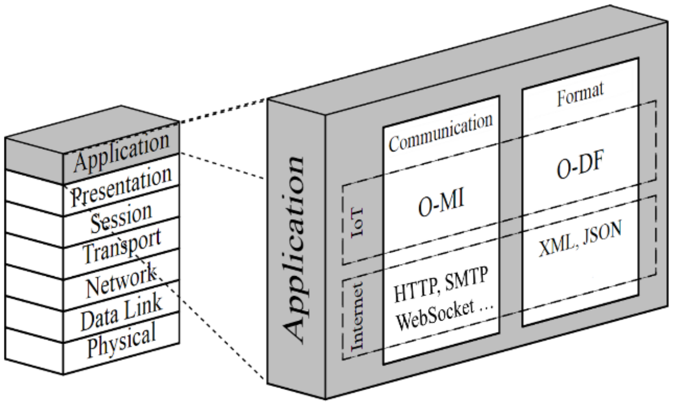
- Introduction
- Open Messaging Standards (O-MI and O-DF)
- O-MI/O-DF Reference Implementation
- How to run O-MI node
- Live demos
- (Scripts and Slides are available at: https://github.com/AaltoAsia/Dublin-workshop)

Introduction

- This workshop will address the practical aspects of using IoT technology to deliver smart city solutions.
- Goal of the Workshop:
  - Install O-MI Node reference implementation
  - Use an IoT device to send sensor values to your Node
  - Make a data subscription to send value from your Node to our Node
- We will have **live visualization** of all the data coming to our O-MI Node

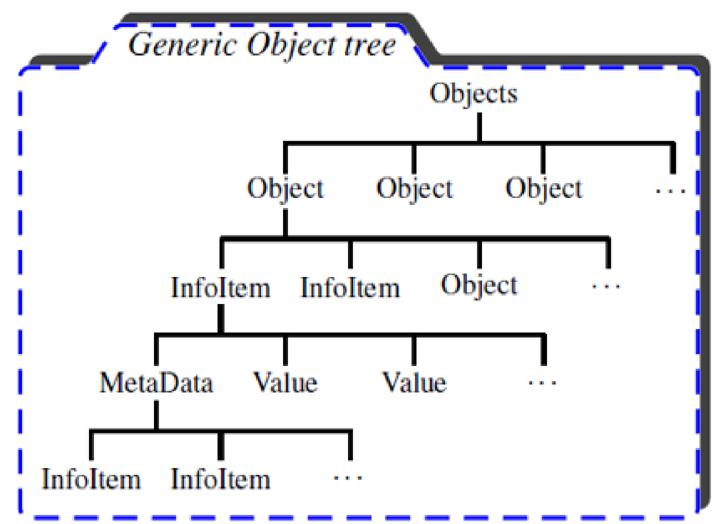
Open Messaging Standards

- Defined by The Open Group as Open Messaging Interface (O-MI) and Open Data Format (O-DF)
- Provide peer-to-peer communication and real-time interaction between devices or information systems
- Capabilities: IoT CRUD (Create, Read, Update, Delete)
- **O-MI:** Provides a framework to publish and consume real-time information
- **O-DF:** Represents a data payload



Open Data Format (O-DF)

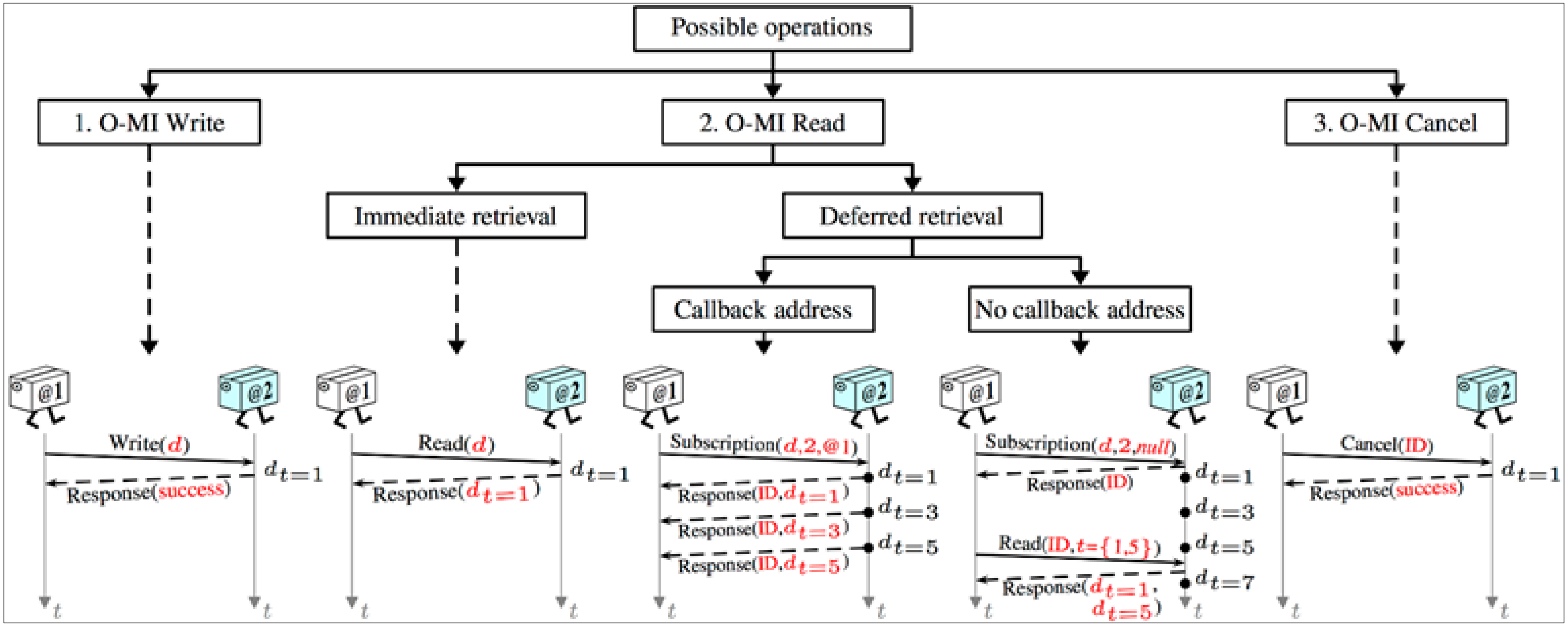
- Defined as a simple ontology and specified using XML Schema
- Provides structure and mechanism to annotate data for information exchange
- Generic enough for representing "any" object and information



Open Message Interface (O-MI)

- Enables communication between heterogeneous devices and information systems
- O-MI node can act both as a "server" and as a "client"
- O-MI Properties:
  - Self-contained messages
  - Protocol-agnostic messages
  - Different payload formats
  - Specifying time-to-live
  - Publication and discovery of new services and metadata
  - Subscription of data or services
- Often used on top of HTTP or Websockets

O-MI Basic Operations



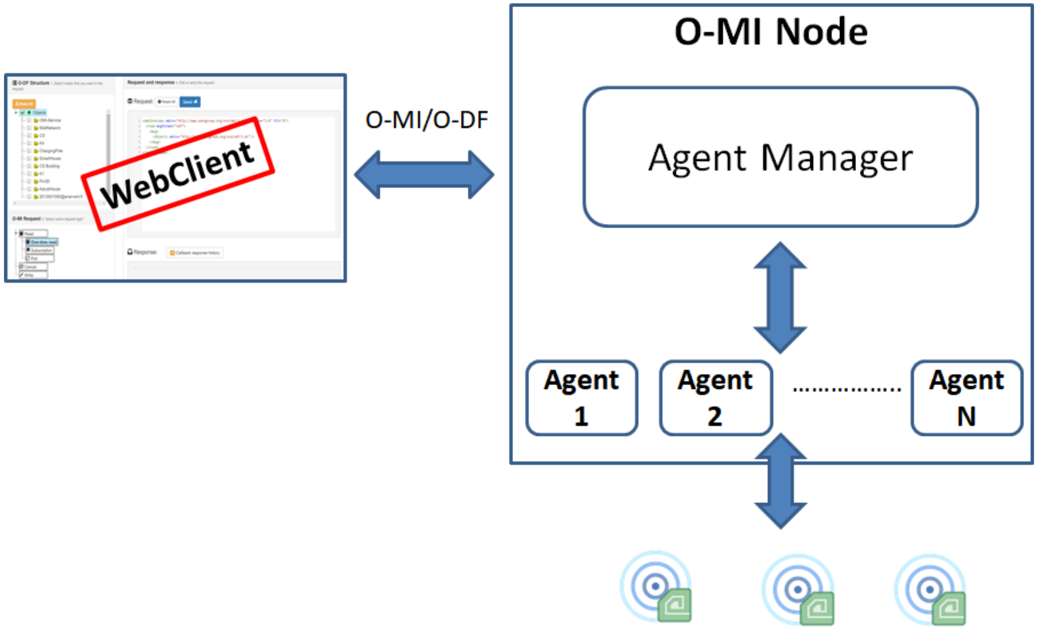
- **Write** information, such as sensor values, setpoints, alerts, etc
- **Read** current and historical information, alerts, other events
- **Subscribe** to information with or without callback
- **Cancel** subscriptions before expiration

An example Write message

```xml
<omiEnvelope xmlns="http://www.opengroup.org/xsd/omi/1.0/" version="1.0" ttl="0">
  <write msgformat="odf">
    <msg>
      <Objects xmlns="http://www.opengroup.org/xsd/odf/1.0/">
        <Object>
          <id>SmartFridge22334411</id>
          <InfoItem class="FridgeTemperatureSetpoint">
            <value>3.5</value>
          </InfoItem>
        </Object>
      </Objects>
    </msg>
  </write>
</omiEnvelope>
```

O-MI/O-DF Reference Implementation

- Open source implementation developed at Aalto University
  - Available at: https://github.com/AaltoAsia/O-MI
- Enable fast deployment of IoT node



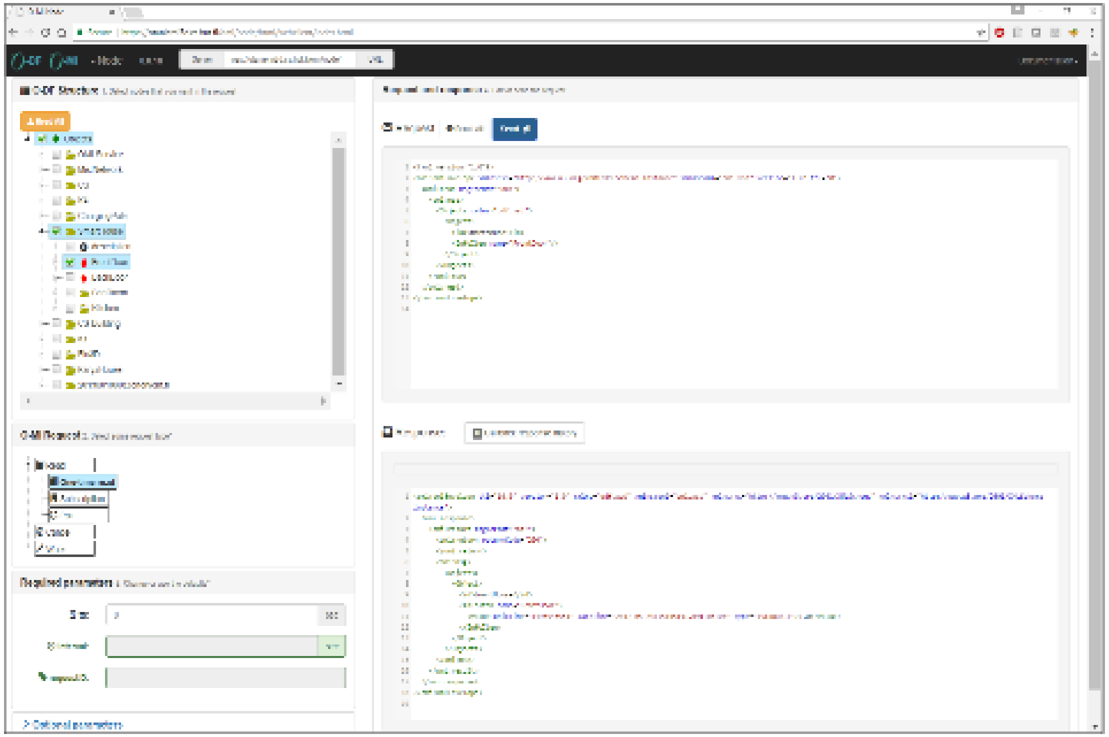
O-MI Node Server

- Main building block of reference implementation
- Supports all O-MI operations, handles user requests, and manages the database
- Agent management system
  - Mechanism to interact programmatically with the core of an O-MI node
  - Can filter and modify incoming O-MI requests
  - Can push and pull sensor data to and from database

Web Client

Aalto University
School of Science
and Technology

- Graphical user interface for the developers to create and test O-MI messages
- Sandbox for testing O-MI operations or features



Wrapper

Aalto University
School of Science
and Technology

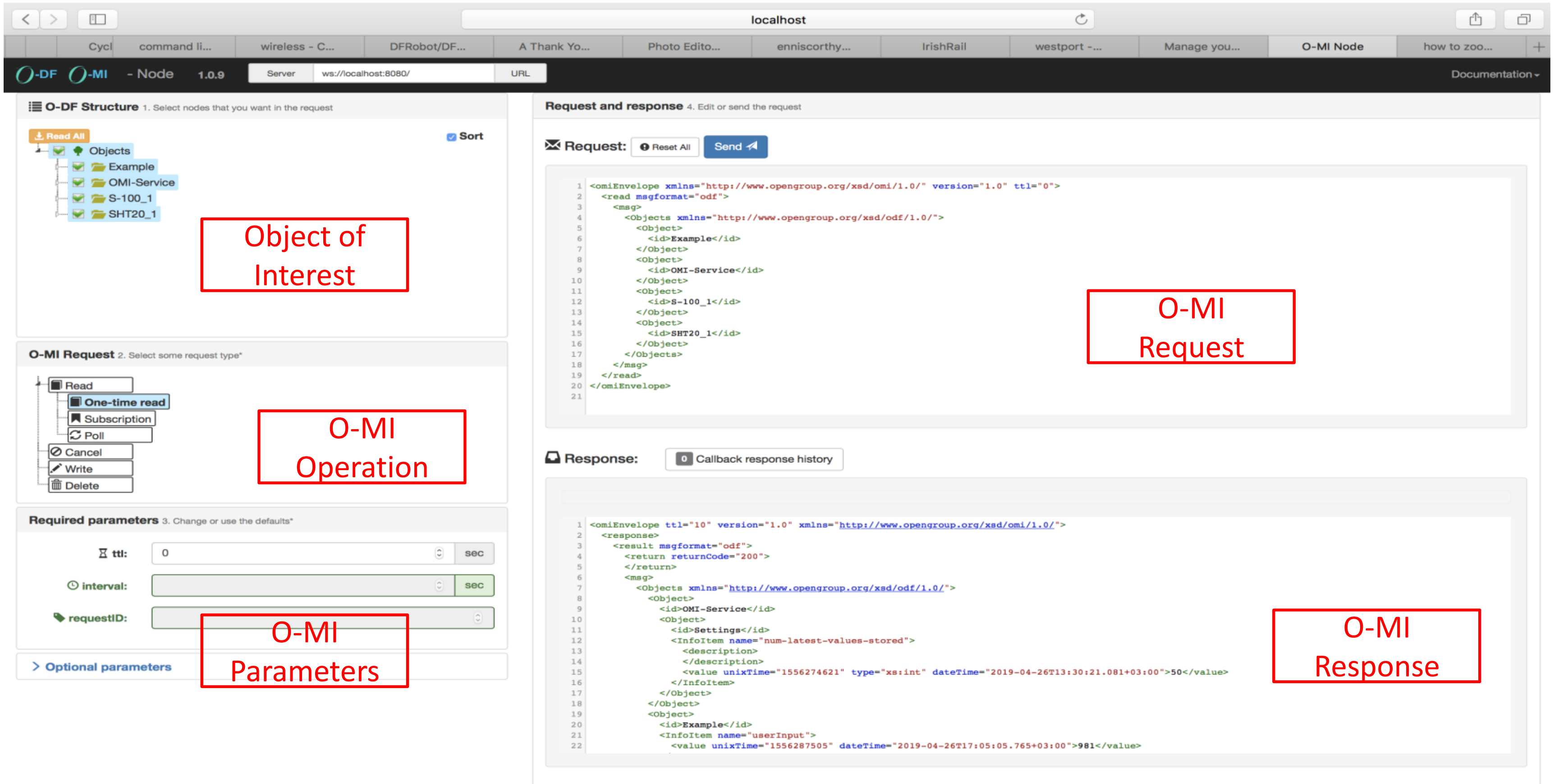
- A piece of code that converts to or from O-MI/O-DF is considered as a “wrapper”
- Combine data from different source or system with different/same protocol into single O-DF structure for publishing
- Encapsulate underlying lower protocol for data consumer
  - e.g. 1-Wire, third party system, KNX etc.
- Wrapper can be written in any programming or scripting language.
  - e.g.: Shell script, Java, Python, C etc.

How to run O-MI node?

- Dependency Java 1.8 (https://www.java.com/en/download/)
- Run pre-compiled node:
  1. Download the latest release version (.zip/.tar) and unpack it. Available: https://github.com/AaltoAsia/O-MI/releases
  2. Go to the <Extracted folder>/bin and run o-mi-node.bat (for Windows) or o-mi-node (for Unix/Mac)
  3. The server can now be accessed with URL http://localhost:8080/
  4. To allow **write** requests from other machines you need to modify file "<Extracted folder>/conf/ application.conf".
     - Change the setting "**allowRequestTypesForAll"**, add "**write**" to the list, such that the result looks like this:
     - allowRequestTypesForAll = ["read", "cancel", "call", "**write**"]
  5. Restart the O-MI Node
- If you want to compile from source code: Follow instructions on the Github Readme

O-MI/O-DF Sandbox

Aalto University
School of Science
and Technology

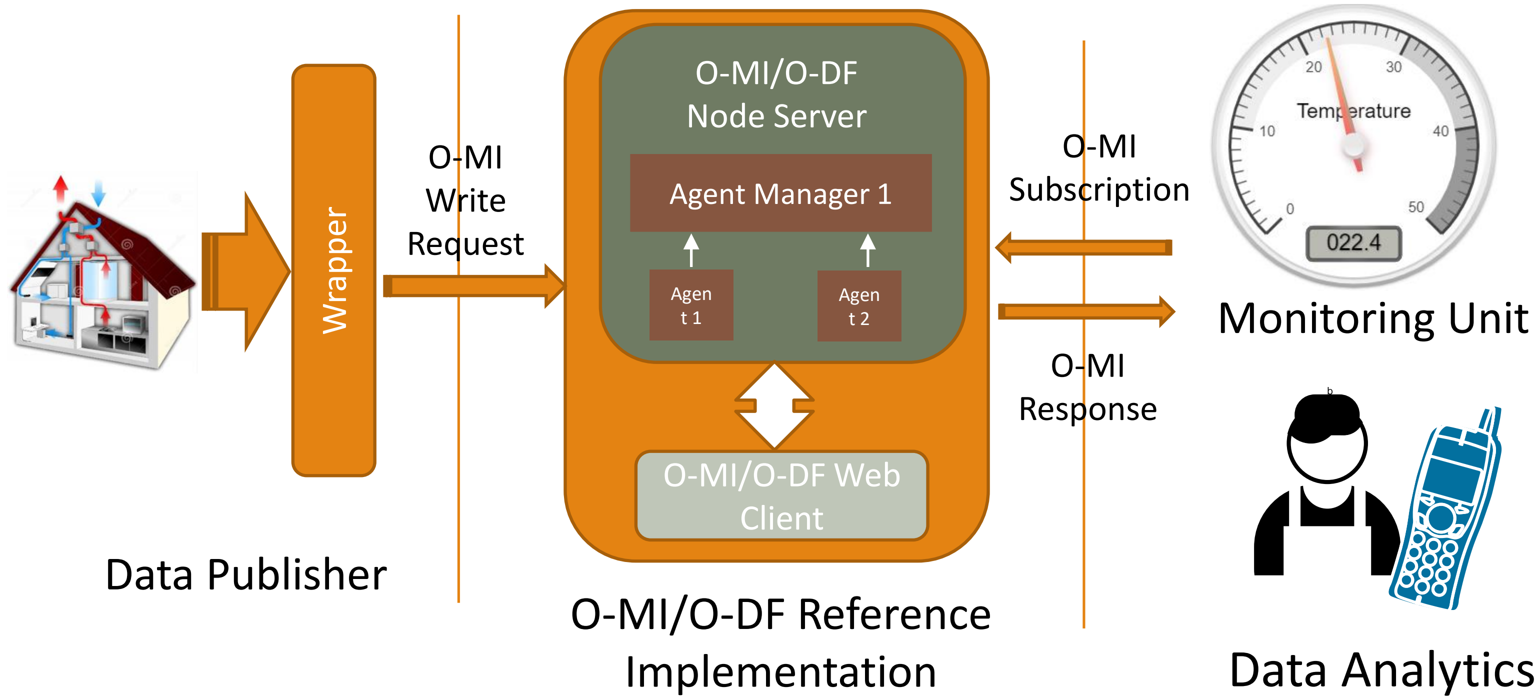


Steps to publish data with reference implementation

- Step 1: Deploy reference implementation
- Step 2: Develop wrapper for getting and publishing data
- Step 3: Send O-MI write request to create or update data periodically using wrapper
- Step 4: Use Web Client to test other O-MI/O-DF functionality such as one time read, subscription, delete etc.

Note: Other system can also read/subscribe published data by sending read/subscription request.

Case Study: Smart Home



Smart Home Scenario Description

Aalto University
School of Science and Technology

- Smart Home equipped with
  - Temperature and Humidity sensor (SHT-20 and 1-wire)
  - CO2 sensor (S-100)
- Wrapper to publish smart home data to O-MI node
  - Read sensor value
  - Translate it to O-DF
  - Put it in O-MI Write request
  - Send with HTTP POST request
- Developer can test the system using web-client
- Data consumer (Monitoring unit) might be interested in these sensor values for analyzing the required object values. It can subscribe to the objects of interest

Hardware and Sensors Provided

- Raspberry Pi boards (Model 1 and 3)
- ESP8266 WiFi development modules
- SHT20 Temperature and humidity sensor
- S-100 CO2 sensor
- 1-Wire temperature and humidity sensor

Live Demo

Implementation
Using Raspberry Pi (RPi)

1-Wire Sensor Installation

1. sudo raspi-config -> Interfacing options
   - enable 1-wire
   - enable i2c protocol
2. Add ”i2c-bcm2708” and ”i2c-dev” modules (on separate lines) in /etc/modules
   - sudo nano /etc/modules
3. sudo reboot
4. Verify that i2c bus is enabled
   - sudo i2cdetect –y 1
   - You should get a table as result

OWFS Installation

1. Firstly install some neccesary and useful packages
   - sudo apt-get update
   - sudo apt-get install automake autoconf autotools-dev gcc-4.7 libavahi-client-dev libtool libusb-dev libusb-1.0-0-dev libfuse-dev swig python2.7-dev i2c-tools
2. Download the latest version of OWFS
   - cd /usr/src
   - sudo wget -O owfs-latest.tgz http://sourceforge.net/projects/owfs/files/latest/download
3. Unpack OWFS
   - sudo tar xzvf owfs-3.0p0.tgz
4. Go to the directory and configure OWFS
   - sudo ./configure

OWFS Installation

5. Compile OWFS
   - sudo make (sudo make -j 4, if RPi-3)
   - sudo make install
6. Create mountpoint for 1-wire
   - sudo mkdir /mnt/1wire
7. To access the 1-wire without root privileges:
   - sudo nano /etc/fuse.conf
   - Change #user_allow_other to user_allow_other

Connecting to Raspberry Pi with SSH

Aalto University
School of Science
and Technology

- Connect to our WiFi “TP-LINK_C5E618”
- For Windows:
  - Use putty (https://www.putty.org/)
- Connect to RPi:
  - ssh pi@ipAddress
- ssh (to RPi-3)
  - User: pi
  - Password: Qwerty123
- ssh (to RPi-1)
  - User: pi
  - Password: raspberry

RPi ID	IP address
R1	192.168.1.10
R2	192.168.1.11
R3	192.168.1.12
R4	192.168.1.13
R5	192.168.1.14
M1	192.168.1.15
W1	192.168.1.17
W2	192.168.1.16

Start 1-wire (Start from here with RPi)

1. Start OWFS
   - sudo /opt/owfs/bin/owfs -u /dev/ttyUSB0 --allow_other /mnt/1wire/
2. Check the contents of 1-wire
   - ls /mnt/1wire/

     (You will find the directory starting with 26. which contains the sensors data (or 2 directories with 26. if using two 1-wire sensors in chain))
3. Read the sensor values from the directory starting with ”26.”
   - cat /mnt/1wire/<26…>/temperature
   - cat /mnt/1wire/<26…>/humidity
4. Download the scripts on Raspberry Pi and go to the cloned directory:
   - git clone https://github.com/AaltoAsia/Dublin-workshop.git
   - cd Dublin-workshop

Start 1-wire

- 'omi-write.sh' and 'create-odf.sh' scripts on GitHub is used to send 1-wire data to O-MI node
- In the "create-odf.sh" script:
  - Type "nano create-odf.sh" to edit this script file
  - Update the Object id to some unique name. Change the name 'OneWireSensor' (appeared in two places) to your own defined name.
  - Update the 1-wire sensor id in the following line:

```
"cat /mnt/1wire/<sensor-id>/temperature"` `eval "cat /mnt/1wire/<sensor-id>/humidity"`"
```

Place sensor id here starting with 26.

Start 1-wire

- Find the IP address of your laptop (where you run O-MI node):
  - For Mac/Linux: Run “ifconfig” in the terminal and note the IP address of **wlan0** interface:

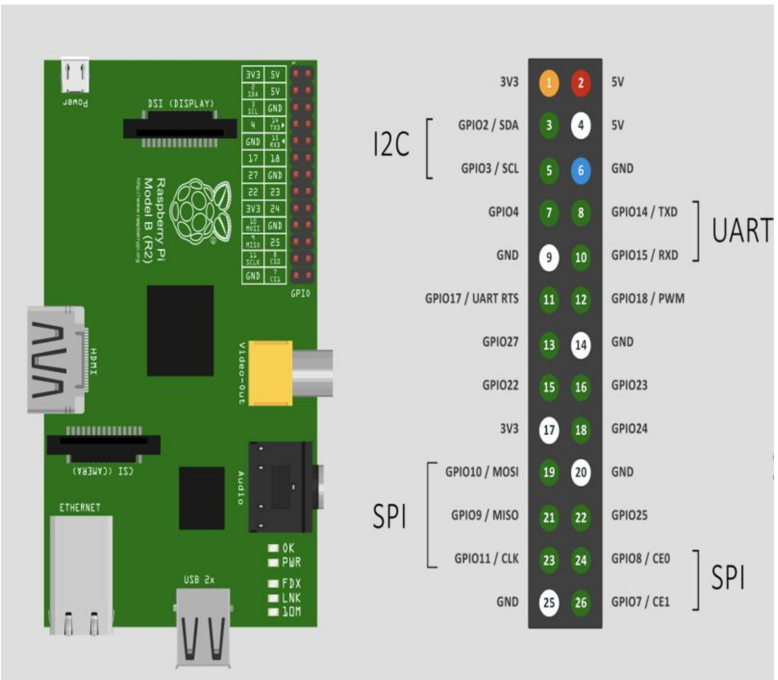
```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.17  netmask 255.255.255.0  broadcast 1
        inet6 fe80::48a0:8a24:9050:634c  prefixlen 64  scope
        ether b8:27:eb:9e:dd:1a  txqueuelen 1000  (Ethernet)
```

  - For Windows: Run “ipconfig” in the command prompt.

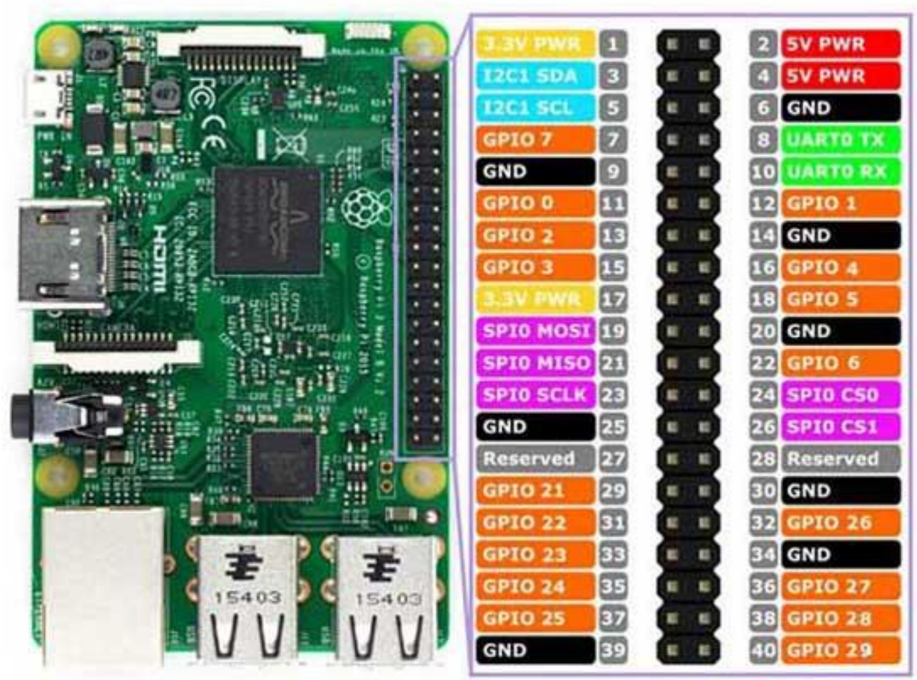
- In the “omi-write.sh” script:
  - Change the address of O-MI node to http://yourIP:8080 in the following line at the end of script:
    - “curl_cmd=$(curl --data "$(omiWrite)" "http://myhost:port")”

- Finally, run the script
  - ./omi-write.sh

Raspberry Pi Pinout

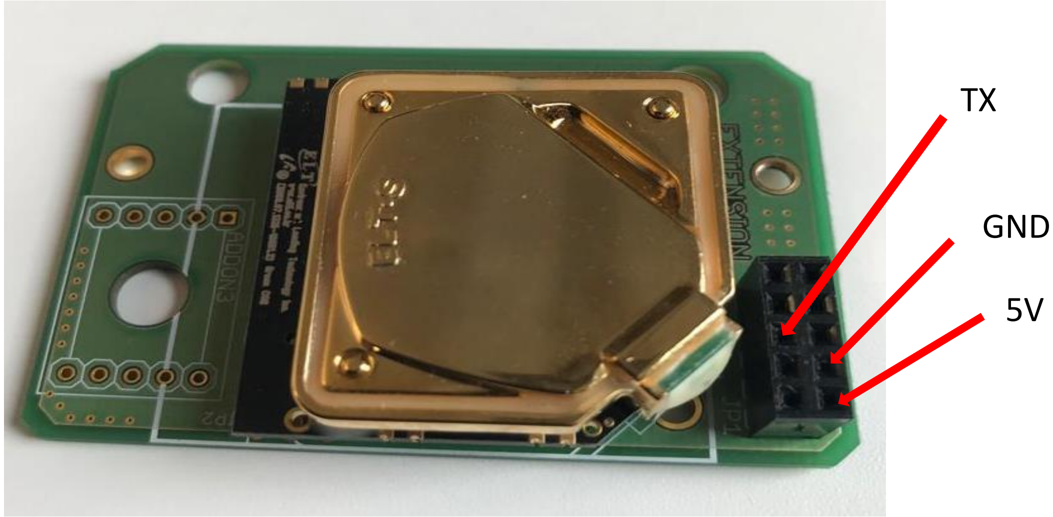


Raspberry Pi 1



Raspberry Pi 3

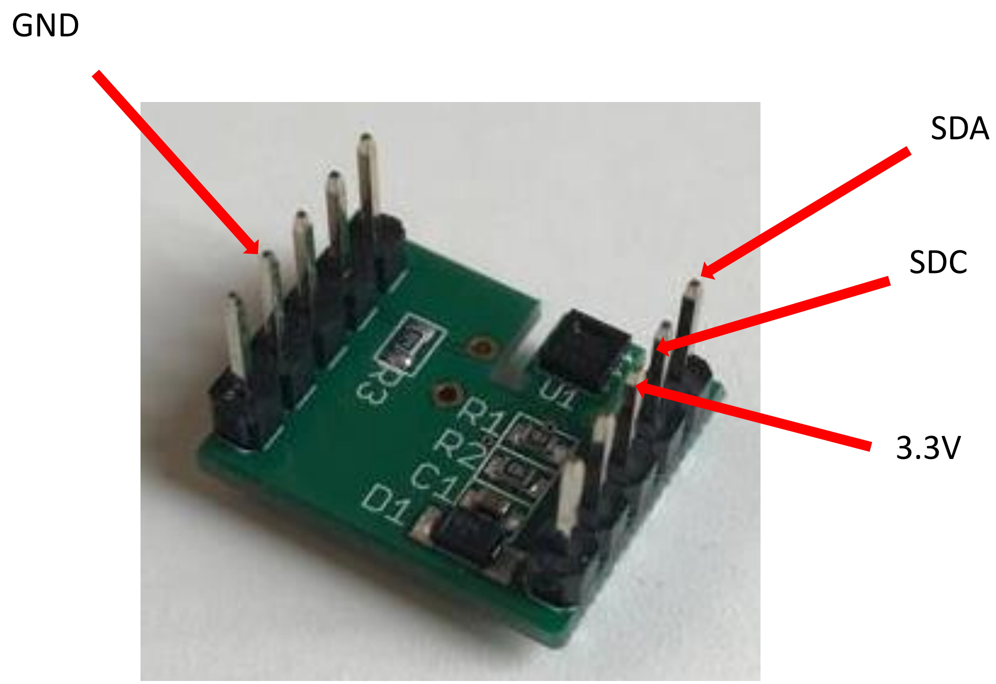
CO_2 Sensor (Model: S-100)



- Connect jumper cables:
  - Sensor TX -> RPi RX
  - Sensor GND -> RPi GND
  - Sensor 5V -> RPi 5V
- Run './co2-omi-send.py' script

Temperature and Humidity Sensor (Model: SHT-20)

# ?	* ?
+ GND	* ?
*?	* 3v3
*?	* SDC
*?	* SDA



Temperature and Humidity Sensor (Model: SHT-20)

- Install the sensor python library
  - sudo apt-get update
  - sudo pip3 install sensor
- Connect the jumper cables:
  - Sensor SDC -> RPi SDC
  - Sensor SDA -> RPi SDA
  - Sensor GND -> RPi GND
  - Sensor 3V -> RPi 3V
- Read the temperature and humidity value (run the provided python script './ht-omi-send.py' or 'python3 ht-omi-send.py')
- Important steps:
  - Do change the address of O-MI node (http://yourIP:8080)
  - Also, please change the Object id to some unique name in the O-MI XML structure

Live Demo

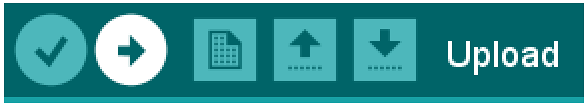
Implementation Using ESP8266

ESP8266 Arduino Configuration

- The ESP8266 is a very low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability.
- Arduino is an open-source electronics platform based on easy-to-use hardware and software.

1. Follow the instructions on the link (https://github.com/esp8266/Arduino) to download Arduino and ESP core
2. Start Arduino IDE and select the following board from Tools -> Board
   - LOLIN (WEMOS) D1 R2 and Mini
3. On MAC you might need a driver:
   - https://github.com/adrianmihalko/ch340g-ch34g-ch34x-mac-os-x-driver

ESP8266 Arduino Configuration

4. Select port using Tools -> Port
5. Open the required script
6. Open create_odf tab and change <id> of the Object to something unique and change the url to your O-MI Node (http://yourIP:8080/), also make sure that both devices are on same wifi network
7. Press Upload button 

8. If it fails you might have wrong port or too high upload speed
9. View the output through Tools -> Serial Monitor

SHT-20 Sensor

Aalto University
School of Science
and Technology

- Download the library for SHT20 from the given link
  - https://github.com/DFRobot/DFRobot_SHT20
- Use Jumper cables to connect the required sensor ports (SDA, SDC, Voltage, and GND)
- Use the provided code in the folder 'temp_humi_test'
- Add the IP address of the receiving O-MI node and change the name of O-DF Object id in the code
- Tools -> Serial Monitor
  - Select baud rate as 115200

S-100 Sensor

- Add the IP address of the receiving O-MI node and change the name of O-DF Object id in the code
- Connect TX of S-100 with RX of ESP8266.
- Upload the code on ESP8266

  Note: Disconnect the RX of ESP8266 while uploading the code and connect it again after the uploading is done.
- Tools -> Serial Monitor
  - Select baud rate as 38400

CPU Temperature (For Linux machine)

- Install some required packages
  - sudo apt-get update
  - sudo apt-get install lm-sensors
  - sudo sensors-detect
- Start the service
  - sudo service kmod start
- Read the temperature and send the value to O-MI node using the provided script.
  - while true; do sensors -u | awk '/temp1_input/ {print $2}' | ./omi-send-linux.py; sleep 2s; done

CPU Temperature (For MAC machine)

Aalto University
School of Science
and Technology

- Download latest version of python: (if not already installed)
  https://www.python.org/downloads/release/python-373/
- Download pip for python3 and other required packages:
  - curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  - python3 get-pip.py
  - pip3 install requests
- Reading CPU Temperature
  - Download source code from https://github.com/lavoiesl/osx-cpu-temp
  - Unzip the downloaded folder and 'cd' to that directory
  - Run make
  - Copy omi-send-mac.py to the same directory
  - while true; do ./osx-cpu-temp | ./omi-send-mac.py; sleep 2s; done

Create a subscription for our visualization Node

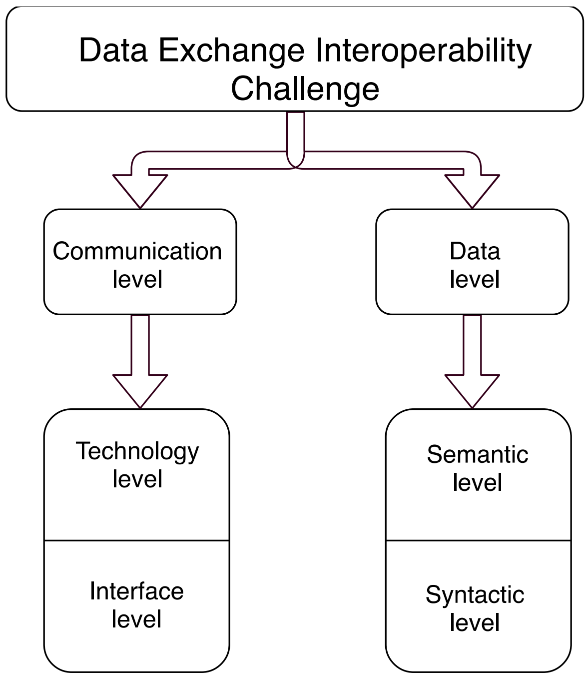
1. Go to http://localhost:8080/ and go to ”O-MI Test Client WebApp”
2. Check that your data have been received correctly with a Read request, after that, continue with subscription:
3. Select ”Objects” from the O-DF tree
4. Select ”Subscription” request
5. Open ”Optional parameters”
6. Put address of the visualization node to ”callback” field
7. Press ”Send” and see if your value shows on the visualization (after it changes next time)


Thank you

Extra slides

Interoperability

- Interoperability Challenge
  - Communication Interoperability
    - Infrastructure and mechanism
    - Technology Level
      - e.g. TCP/IP
    - Interface Level
      - Means of communication
      - e.g. O-MI, HTTP
  - Data Level Interoperability
    - Syntactic Level
      - Common Data Format
      - e.g. O-DF, CSV
    - Semantic Level
      - Shared Meaning
      - e.g. Mobivoc



Raspberry Pi setup

Aalto University
School of Science
and Technology

Operating System Installation:

1. Download and unzip Soft-float Debian “Wheezy" Operating System for Raspberry pi from http://www.raspberrypi.org/downloads
2. For flashing the Wheezy Operating System in SD card of the Raspberry Pi download and unzip Win32DiskImager.exe
from http://sourceforge.net/projects/win32diskimager/
3. Run Win32DiskImager.exe , select the .img file unzipped from step 1, select the SD card and click Write.
4. Eject SD card safely from writer and insert it into raspberry.
5. Insert SD card in raspberry PI and connect the necessary IO devices (mouse, Keyboard and monitor) and Internet cable. start it by supplying the power.

Java Installation

Aalto University
School of Science
and Technology

1. Login into raspberry pi using default credential (username=pi and password= raspberry)
2. sudo apt-get update
3. sudo apt-get install openjdk-7-jdk

Writing data to O-MI Node

- Add the sensor objects while creating data format

```
Object="OneWireSensor"
OneWireSensor_Object="Sensor1"
Sensor1_InfoItems="Temperature Humidity"
Sensor1_values="`eval "cat /mnt/1wire/26.0F85CB010000/temperature"` `eval "cat /mnt/1wire/26.0F85CB010000/humidity"`"
```

- Add the IP address of the receiver O-MI node in write message

```
# Send write message

#curl_cmd=$(curl --data "$(omiWrite)" "http://192.168.1.100:8080")
```

(The scripts have been created for your reference)

CO_2 Sensor (Model: S-100)

- Getting the value of sensor (baud speed is 38400)

```
miniterm.py /dev/serial0 38400
```

Raspberry Pi 1

```
miniterm.py /dev/ttyS0 38400
```

Raspberry Pi 3

- Add the IP address of the receiver O-MI Node in OMI Node write script

```
requests.post("http://192.168.1.103:8080"
```

(The script for sending the data has already been created for your reference)

- Sending the value to O-MI Node

```
miniterm.py /dev/serial0 38400 | ./omi-sent.py
```

Raspberry Pi 1

```
miniterm.py /dev/ttyS0 38400 | ./omi-sent.py
```

Raspberry Pi 3