Internet of Things
2016/2017

Lightweight M2M
based on OMA technical and architecture descriptions
(most pictures are taken from there)

Johan Lukkien
Richard Verhoeven

John Carpenter, 1982
Guiding questions

• What are management concerns of low resource devices?
• How does LWM2M support these, and what model does it provide?
Framework for discussing protocols

- Purpose of the protocol
  - the problems it solves
  - the context it operates in, the place in the stack
- Parties that use it
- Functionality, typical behavior
- Packet format
- Carriers
- Binding to carriers
- Utility for IoT
Purpose and context

• Purpose of LWM2M
  – *application layer protocol* for *management of (constrained) devices*
  – *enables applications to use constrained devices*
  – developed by OMA, the Open Mobile Alliance
  – resembles the Simple Network Management Protocol (SNMP)

• The problems it solves, its context
  – mediation between constrained devices and applications
  – management functions entail
    • *bootstrap*: bootstrapping and upgrading a device
    • *registration*: taking a device into a logical group
    • *management*: monitoring, updating, parameter settings
      – by writing / creating objects inside the device
    • *information*: reading objects inside a device
      – objects can correspond to sensors or actuators
  – standardizing *what is communicated* to constrained devices
  – security (management thereof)
Use cases from OMA

- **Asset management**
  - manage many small devices collectively

- **Examples**
  - street lights
  - air conditioning
  - fleet of vehicles

From: OMA Lightweight M2M Requirements Documet
Terminology

- LWM2M is called an *enabler*
  - since it leaves the exact way of organizing the management open
  - enabler also refers to the software components that make this happen

- **Client**: the low-resource device
  (has both CoAP client and server roles, e.g. CoAP origin *server*)

- **Server**: the manager of several low-resource devices (mostly a CoAP *client*, but sometimes CoAP server)

- **Bootstrap server**: a server that initializes a constrained device

- Four interfaces (CoAP services)

---

From: OMA LWM2M Technical Specification
Overview, place in the stack

From: OMA LWM2M Technical Specification

From: OMA LWM2M Architecture document

30-Nov-16
Deployment options (#1)

- LWM2M Server manages a collection of LWM2M clients
  - could be several different managerial domains
  - hence, clients need to develop an association with the server (security)

- Server provides M2M applications access to LWM2M clients
  - applications considered to reside with the server, i.e., owned together
    • meaning that combined development is possible
  - server represents the LWM2M clients in new services, e.g. aggregation
  - server acts as mediator towards LWM2M clients
    • structure resembles the Philips HUE organization

- Server plus application provide services to users
Deployment options (#2)

- Server manages a collection of LWM2M clients
  - could be several different managerial domains (hence: association)

- Server provides M2M applications access to LWM2M clients
  - the applications are in a different managerial domain than the server
    - hence, some application development interface needs to be designed, including security considerations
    - clients may have second thoughts on the role of the Network service provider
      - server represent the LWM2M clients in new services, e.g. aggregation
      - server acts as mediator towards LWM2M clients

- Application provides services to users
LWM2M clients can connect to several servers
  - here: each server: #1client

There are several managerial domains now

LWM2M client can remain approachable by third parties (also in previous cases)

From: OMA LWM2M Architecture Document
**Functionality & addressing**

**Functionality**
- Four (service) interfaces
  - bootstrap
  - register
  - device management
  - information
- These can be used, in principle, by different servers
- Three types of entities:
  - LWM2M bootstrap server
  - LWM2M server
  - LWM2M client

**Addressing**
- Clients have names, stored upon registration
- Clients contain objects
  - numbered objects
  - have numbered instances
  - and contain numbered resources
  - example: /0/0/2

---

Johan J. Lukkien, j.j.lukkien@tue.nl  
TU/e Informatica, System Architecture and Networking

30-Nov-16
Bootstrap

• Obtain sufficient information to register at an LWM2M server

• Sequence:
  – Use smartcard if available
  – If that fails, use factory settings
  – Try to register at a (known in this way) LWM2M server
  – If registration fails, wait for Server Initiated Bootstrap
    • Based on client device detection by network provider
  – Else, perform Client Initiated Bootstrap

• Operations: write, delete
Client register

- Specify endpoint client name
  - node1
- Specify objects and instances available in the client
  - </1/1>, </2/1>, ….
- Optionally:
  - Lifetime
  - LWM2M version
  - Binding mode
  - SMS number
- Operations: (de-)register, update mapped to create, update (write), delete
Device Management

- Operates through manipulating (reading, writing, executing) resources

- Write /3/0/1
  - 3 = Object ID (= device object)
  - 0 = Object Instance ID
  - 1 = Resource ID (= string identifying this client)

- The ‘execute’ is a reboot

- Operations
  - Read, Discover, Write, Write Attributes, Execute, Create, Delete
Information Reporting

- Operations:
  - Observe
  - Notify
  - Cancel Observation

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Informatica, System Architecture and Networking
Object and Resource IDs

- Registered with Open Mobile Naming Authority
- Example objects:

<table>
<thead>
<tr>
<th>Object</th>
<th>Object ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWM2M Security</td>
<td>0</td>
</tr>
<tr>
<td>LWM2M Server</td>
<td>1</td>
</tr>
<tr>
<td>Access Control</td>
<td>2</td>
</tr>
<tr>
<td>Device</td>
<td>3</td>
</tr>
<tr>
<td>Connectivity Monitoring</td>
<td>4</td>
</tr>
<tr>
<td>Firmware</td>
<td>5</td>
</tr>
<tr>
<td>Location</td>
<td>6</td>
</tr>
<tr>
<td>Connectivity Statistics</td>
<td>7</td>
</tr>
</tbody>
</table>
Object and Resource Examples

- /3/0/4  Device, Reboot
- /3/0/5  Device, Factory Reset
- /3/0/9  Device, Battery Level
- /5/0/1  Firmware Update, Package
- /5/0/2  Firmware Update, Update
- /6/0/0  Location, Latitude
- /7/0/2  Connectivity Statistics, Tx Data

- IPSO examples (see references)
Object and Resource IDs with the IPSO alliance

- IPSO alliance: IP for Smart Objects
- IPSO Smart Objects Starter Pack
  - Set of 18 smart objects for use with CoAP
  - Based on LWM2M object model
  - Not dependent on full LWM2M framework
    - Use CoAP directly
- IPSO Smart Objects Expansion Pack
  - 16 Common Template sensors
    - Pressure, Power, Distance, ...
  - 6 Special Template sensors
    - Energy, Color, GPS Location, ...
  - 5 Actuators
    - Buzzer, Display, ...
  - 6 Control switch types
    - Up/Down, Push button, Multiple axis joystick, ...
Data Formats

- **Plain text**
  - UTF-8 encoded string

- **Opaque**
  - Sequence of binary octets, like firmware images
  - No structure assumed

- **TLV – Type-Length-Value**
  - Binary representation
  - Multiple values in nested structures
  - Compact, easy to process

- **JSON**
  - JavaScript Object Notation
  - Text-encoding format for key-value pairs
  - Less compact, but ‘human readable’
Data Formats - TLV

- Result of ‘GET /2’: Access Control Lists
  - queries for all objects of type 2
  - returning one object instance in this case

<table>
<thead>
<tr>
<th>TLV</th>
<th>Type Byte</th>
<th>ID Byte(s)</th>
<th>Length Byte(s)</th>
<th>Value</th>
<th>Total Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Control Object Instance 0</td>
<td>0b00 0 01 000</td>
<td>0x00</td>
<td>(17 bytes)</td>
<td>The next 6 rows</td>
<td>3</td>
</tr>
<tr>
<td>Object ID</td>
<td>0b11 0 00 001</td>
<td>0x00</td>
<td>(1 byte)</td>
<td>0x03 [8-bit Integer]</td>
<td>3</td>
</tr>
<tr>
<td>Object Instance ID</td>
<td>0b11 0 00 001</td>
<td>0x01</td>
<td>(1 byte)</td>
<td>0x01 [8-bit Integer]</td>
<td>3</td>
</tr>
<tr>
<td>ACL</td>
<td>0b10 0 00 110</td>
<td>0x02</td>
<td>(6 bytes)</td>
<td>The next 2 rows</td>
<td>2</td>
</tr>
<tr>
<td>ACL [1]</td>
<td>0b01 0 00 001</td>
<td>0x01</td>
<td>(1 byte)</td>
<td>0b11 10 0000</td>
<td>3</td>
</tr>
<tr>
<td>ACL [2]</td>
<td>0b01 0 00 001</td>
<td>0x02</td>
<td>(1 byte)</td>
<td>0b10 00 0000</td>
<td>3</td>
</tr>
<tr>
<td>Access Control Owner</td>
<td>0b11 0 00 001</td>
<td>0x03</td>
<td>(1 byte)</td>
<td>0x01 [8-bit Integer]</td>
<td>3</td>
</tr>
</tbody>
</table>
Data Format - JSON

- Result of ‘GET /3/0’ = instance 0 of Device object (= #3)

- Interpretation examples
  - name 0 (Manufacturer):: stringvalue: “Open Mobile Alliance”
    - i.e. resource number 0 is this string
  - resource 6 (available power source)
    - instance 0: value: 1 (internal battery)
    - instance 1: value: 5 (USB)
Security

• Principles:
  – Authentication of all communication
  – Encrypt all data and protect integrity
  – Access control to object instances and resources

• Implementations
  – UDP channel security
    • DTLS
    • Pre-shared keys
    • Raw public key certificates (optional)
    • X.509 certificates (optional)
    • NoSec (not recommended)
  – SMS channel security
    • DTLS over SMS (device end-point)
    • Secure SMS (smartcard end-point)
## Binding to CoAP

<table>
<thead>
<tr>
<th>Operation</th>
<th>CoAP</th>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Bootstrap</td>
<td>POST</td>
<td>2.04</td>
<td>4.00</td>
</tr>
<tr>
<td>Write</td>
<td>PUT</td>
<td>2.04</td>
<td>4.00</td>
</tr>
<tr>
<td>Delete</td>
<td>DELETE</td>
<td>2.02</td>
<td>4.05</td>
</tr>
<tr>
<td>Register</td>
<td>POST</td>
<td>2.01</td>
<td>4.00, 4.09</td>
</tr>
<tr>
<td>Update</td>
<td>PUT</td>
<td>2.04</td>
<td>4.00, 4.04</td>
</tr>
<tr>
<td>De-register</td>
<td>DELETE</td>
<td>2.02</td>
<td>4.04</td>
</tr>
<tr>
<td>Read, Discover</td>
<td>GET</td>
<td>2.05</td>
<td>4.01, 4.04, 4.05</td>
</tr>
<tr>
<td>Write</td>
<td>PUT / POST</td>
<td>2.04</td>
<td>4.00, 4.01, 4.04, 4.05</td>
</tr>
<tr>
<td>Write Attributes</td>
<td>PUT</td>
<td>2.04</td>
<td>4.00, 4.01, 4.04, 4.05</td>
</tr>
<tr>
<td>Execute</td>
<td>POST</td>
<td>2.04</td>
<td>4.00, 4.01, 4.04, 4.05</td>
</tr>
<tr>
<td>Create</td>
<td>POST</td>
<td>2.01</td>
<td>4.00, 4.01, 4.04, 4.05</td>
</tr>
<tr>
<td>Delete</td>
<td>DELETE</td>
<td>2.02</td>
<td>4.01, 4.04, 4.05</td>
</tr>
<tr>
<td>Observe</td>
<td>GET (with observe)</td>
<td>2.05</td>
<td>4.04, 4.05</td>
</tr>
<tr>
<td>Cancel Observation</td>
<td>Reset message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notify</td>
<td>Asynchronous response</td>
<td>2.04</td>
<td></td>
</tr>
</tbody>
</table>
Guiding questions

• What are management concerns of low resource devices?
• How does LWM2M support these, and what model does it provide?
  – We understood how to manage devices
  – Building applications on top of servers has not been explained!
Framework for discussing protocols

• Purpose of the protocol
  – the problems it solves
  – the context it operates in, the place in the stack

• Parties that use it

• Functionality, typical behavior

• Packet format

• Carriers

• Binding to carriers

• Utility for IoT