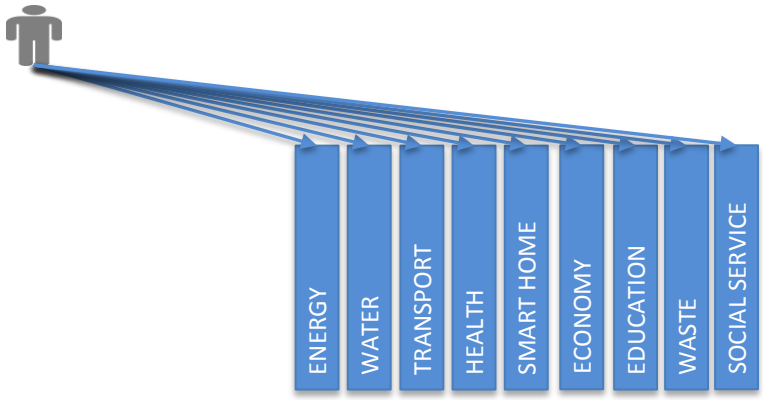
O-MI/O-DF Standards

Manik Madhikermi

Aalto University

Introduction

- IoT and System of Systems (SoS) is expanding.
- Data driven business model.
- Data became important corporate assets.
- Most of companies are creating IoT architecture for data collection resulting vertical silos
- Increasing need of cross domain information exchange for better product or services



Several domains will co-exist and need to interoperate

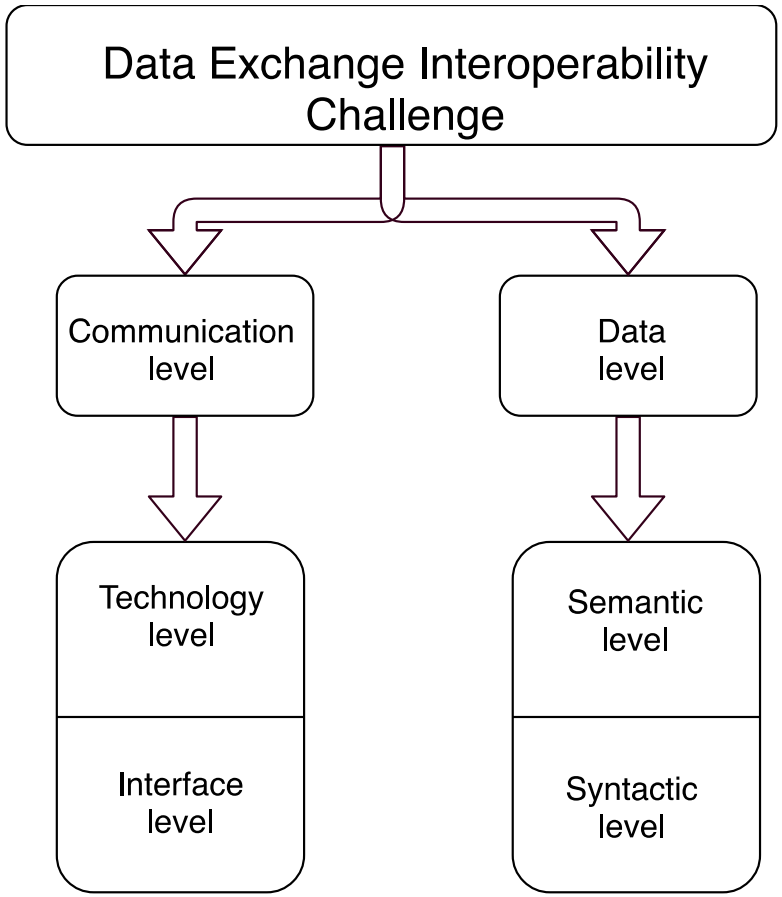
Current Challenges

- Fragmented IoT architectures resulting vertical silos.
- Solutions exist only for specific vertical silos.
- No coherent approaches towards the IoT.
- Vertical silos with variations in technologies and propritery standards
- Standards are domain or lifecycle specific.
- Little or none cross-domain re-use of technology and exchange of knowledge.
- In essence, we have only Intranets of Things.

Vertical Silos prevent the creation of cross-industry, cross-platform and cross-organizational services due to their lack of interoperability and openness

Interoperability

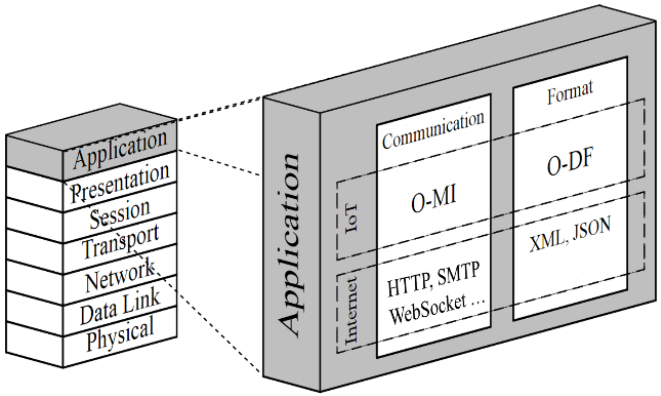
- Interoperability Challenge
  - Communication Interoperability
    - Infrastructure and mechanism
    - Technology Level
      - e.g. TCP/IP
    - Interface Level
      - Means of communication
      - e.g. O-MI, HTTP
  - Data Level Interoperability
    - Syntactic Level
      - Common Data Format
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



Open Messaging Standard

- Application Level Protocol
- Enable peer-to-peer near real-time communication between devices or systems.
- Capabilities: IoT CRUD (Create, Read, Update, Delete)
- Consist of two standards
  - Open Data Format(O-DF)
    - Data payload for IoT applications.
  - Open Message Interface(O-MI)
    - publish and consume real-time information between systems or devices.

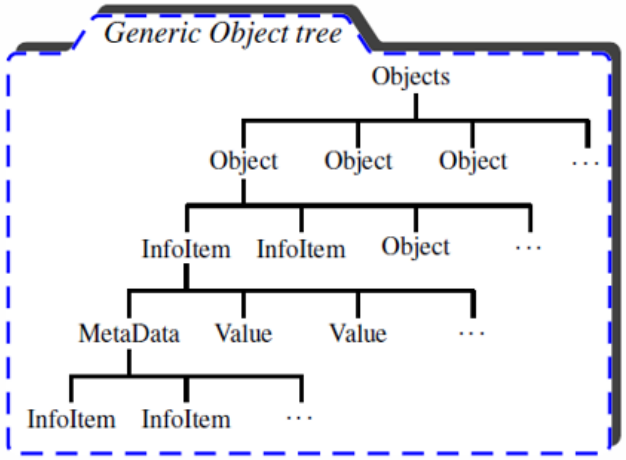
.



Open Data Format (O-DF)

- The Open Group standard for data format.
- Defined as a simple ontology and specified using XML Schema.
- Provides structure and mechanism to annotate data for information exchange.
- Enables requesting and sending published data between systems.
- Generic enough for representing "any" object and information.
- Allows RESTful queries.

.



Open Message Interface (O-MI)

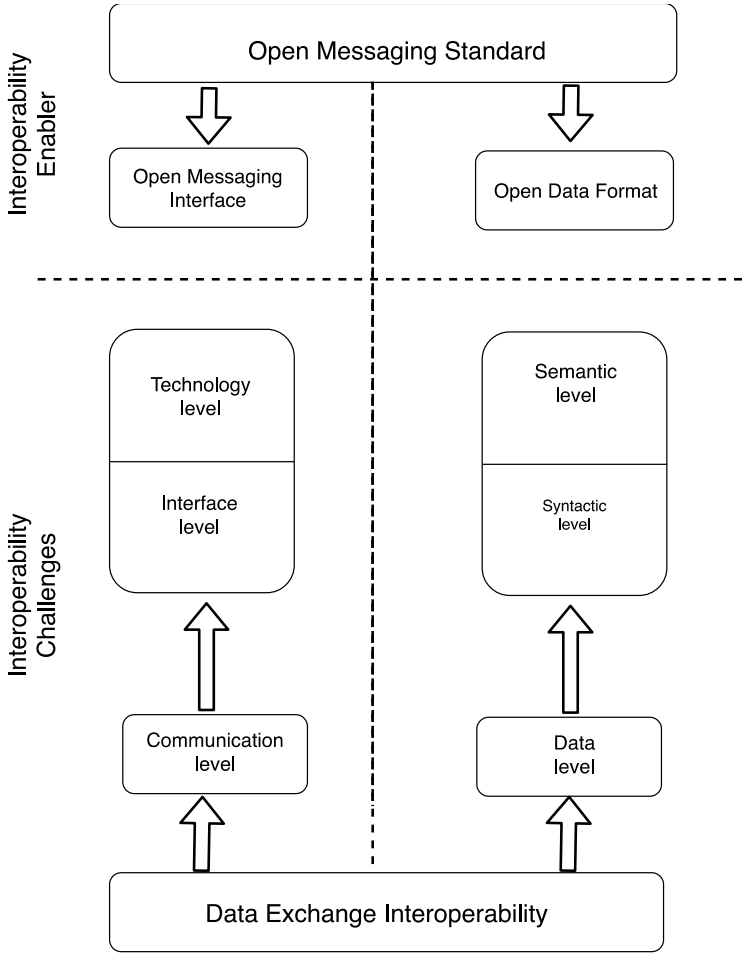
- Enables communication between heterogeneous devices and information systems.
- O-MI Properties:
  - Self-contained messages
  - Protocol-agnostic messages
  - Different payload formats
  - Specifying time to live
  - Publication and discovery of new services and metadata
  - Subscription of data or services
- O-MI Operations
  - Write:
    - write information, such as sensor values, setpoints, alerts, etc.
  - Read:
    - get current and historical information, alerts, other events.
    - subscribes to information with or without callback
  - Cancel:
    - cancel subscriptions before TTL expiration

Example O-MI/O-DF Message

```
<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>
```

Take Away

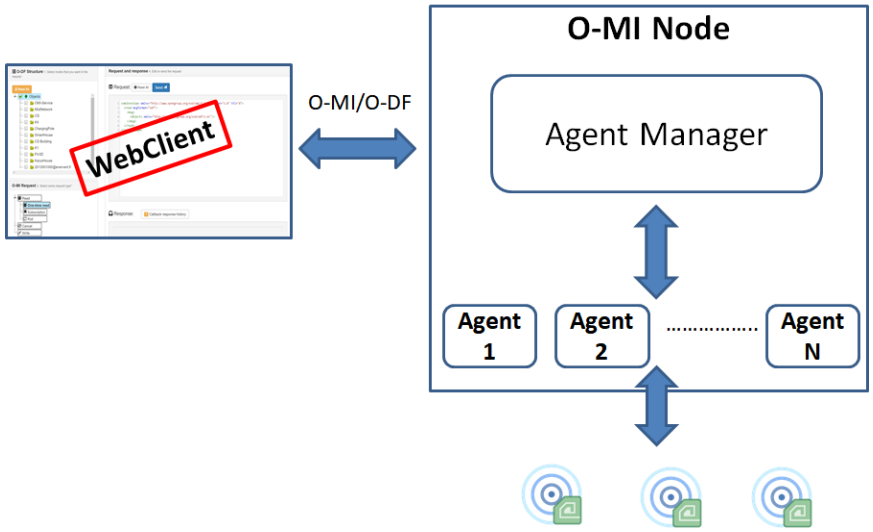
- Open messaging standards
- Interoperability Challenge
  - Communication Level
    - O-MI
      - Provides Interface for data/information exchange
  - Data Level
    - O-DF
      - Provides structure and mechanism to annotate data



REFERENCE IMPLEMENTATION

O-MI/O-DF Reference Implementation

- Open source implementation of O-MI/O-DF Standards
  - Download: https://github.com/AaltoAsia/O-MI
- Developed at Aalto University
- Enable fast deployment of IoT node for peer-to-peer communication
- Components of reference implementation
  - O-MI Node Server (API endpoint)
  - Web Client



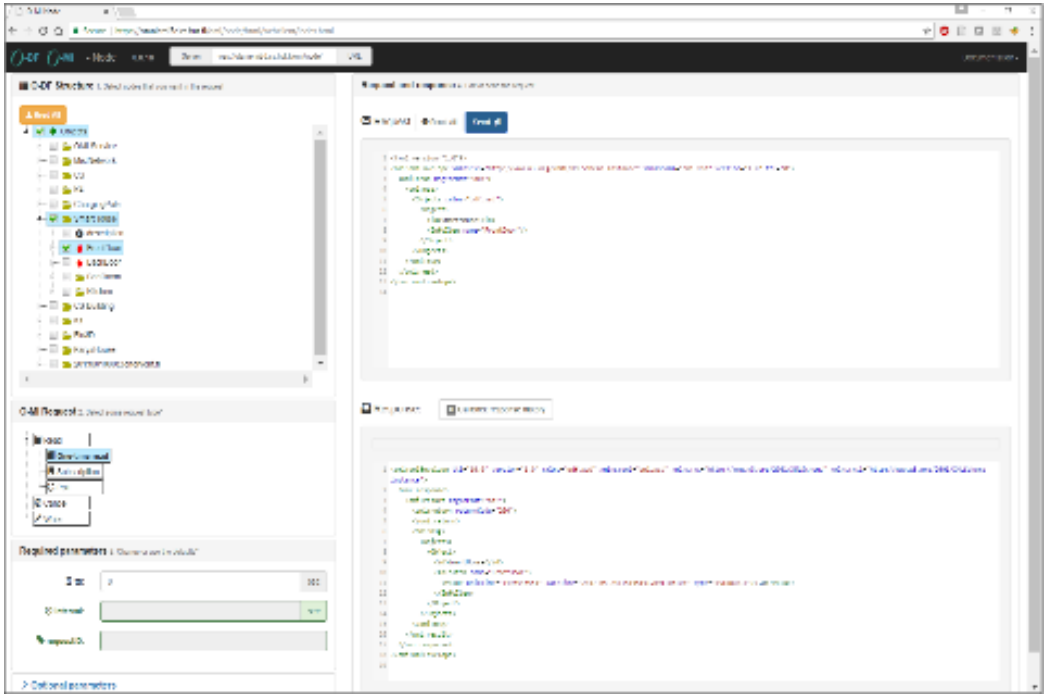
.

O-MI Node Server

- Main building block reference implementation.
- Supports all O-MI operations
- Handle user requests and manages database.
- Agent management system
  - mechanism to interact programmatically with the core of an O-MI node
  - Push and pull sensor data to and from database
  - External Agents
    - Communication between external agents and the O-MI node use a plain TCP socket
    - works by creating O-DF formatted xml data gathered from sensors and sending it to the configured port via a keepalive TCP connection
    - can send O-DF formatted xml data through the open connection
  - Internal Agents
    - Internal agents are extended threads
    - agents listed in the configuration file

O-MI Node Server

- Graphical user interface for the end-users to create and test O-MI messages
- Supports automatic creation of O-MI/O-DF messages
- Sandbox for testing O-MI operations or features.



.

QUICK DEPLOYMENT GUIDE

Deployment Steps

Dependency Java 1.8

- Step 1: Download latest zip file from https://github.com/AaltoAsia/O-MI/releases/
- Step 2: Extract the zip file and navigate to the /bin directory
- Step 3: Run O-MI Node run the corresponding startup script from the bin directory
  - For Windows: bin/o-mi-node.bat
  - For Unix and Mac: bin/o-mi-node
- Step 4: By default application will start at http://localhost:8080/.
- Note: If you want to change configuration such as port it can be configured in /conf/application.conf

Reference configuration and documentation can be found at:

https://github.com/AaltoAsia/O-MI/blob/master/O-MI-Node/src/main/resources/reference.conf

Thank You