ECHONET Lite and/or Web API as Web of Things

Van Cu **PHAM**

Center for Digitalization Endeavors

Japan Advanced Institute of Science and Technology (JAIST)

cupham@jaist.ac.jp

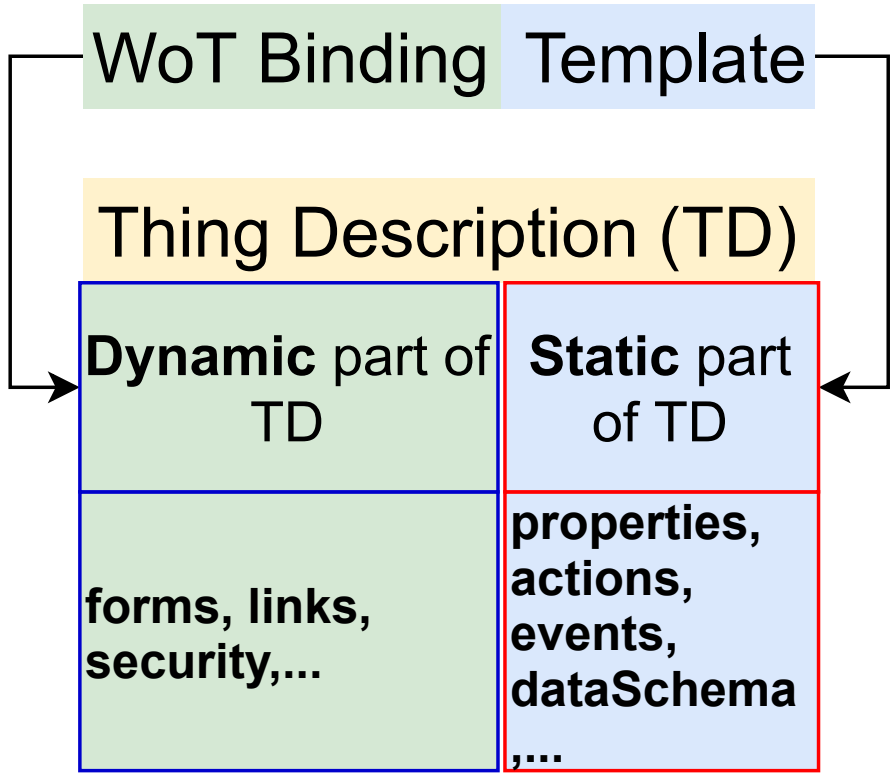
June 16, 2021

Outline

1.1 WoT Binding Template Overview

TODO:

1. **MRA** (DD) → WoT **Thing Description** Mapping (**Static** part)
2. Use WoT Scripting API to add **Dynamic** part at **run-time**
   - Library: **node-wot**
   - Self-Implementation
3. ECHONET Lite-WoT Gateway/Proxy Implementation



Pattern 1: ECHONET Lite <-> **EL-WoT Gateway** <-> WoT Client

Pattern 2: EL WebAPI <-> **EL-WoT Proxy** <-> WoT Client

Pattern 3: **EL-WoT Binding Proxy** + EL WebAPI <-> WoT Client

1.2 ECHONET Lite WoT Cooperation Overview

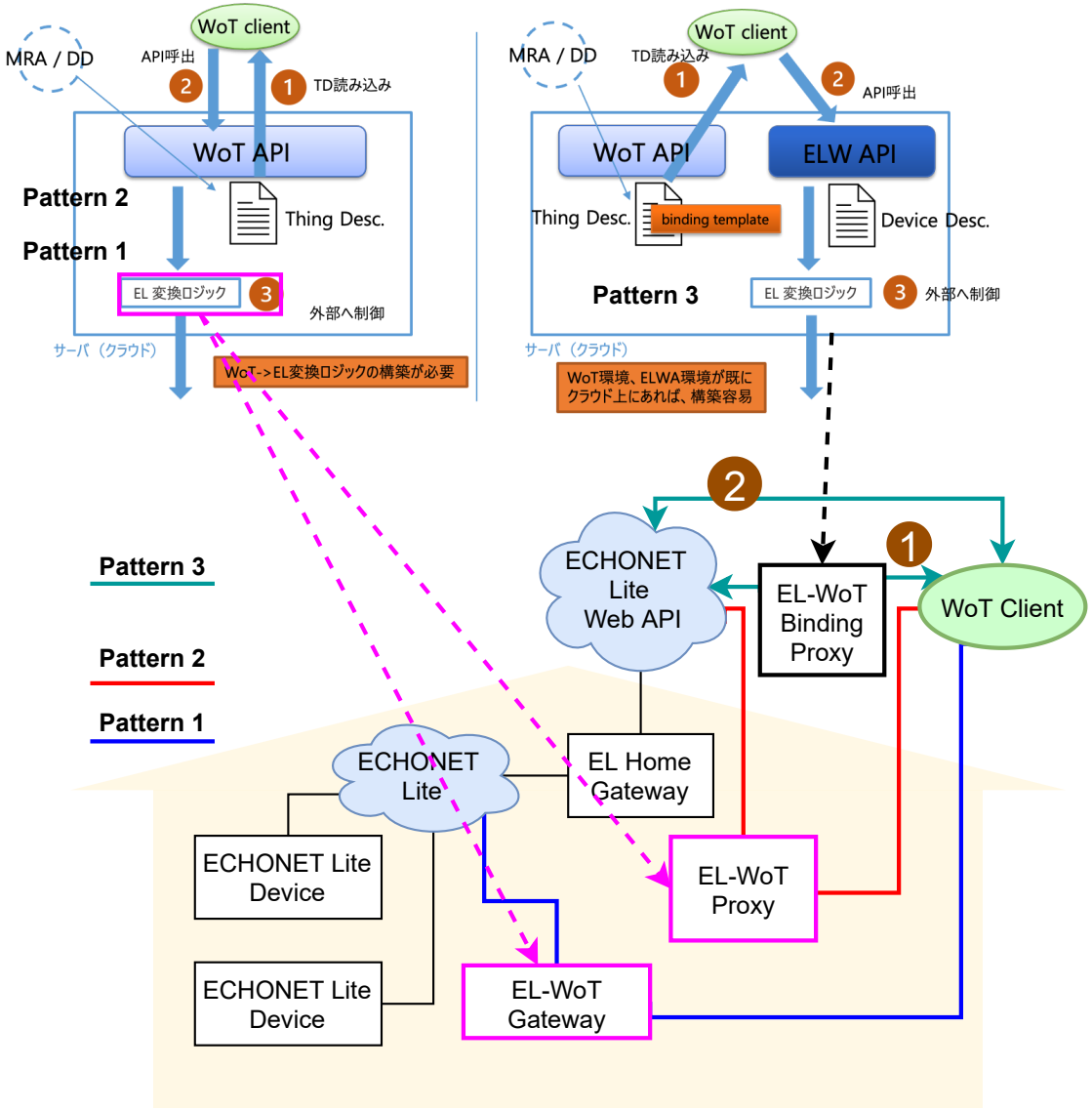
Pattern 1 and **Pattern 2** are similar

- WoT Clients talk to a **Proxy/Gateway** → No need to adjust ELW API and WoT API

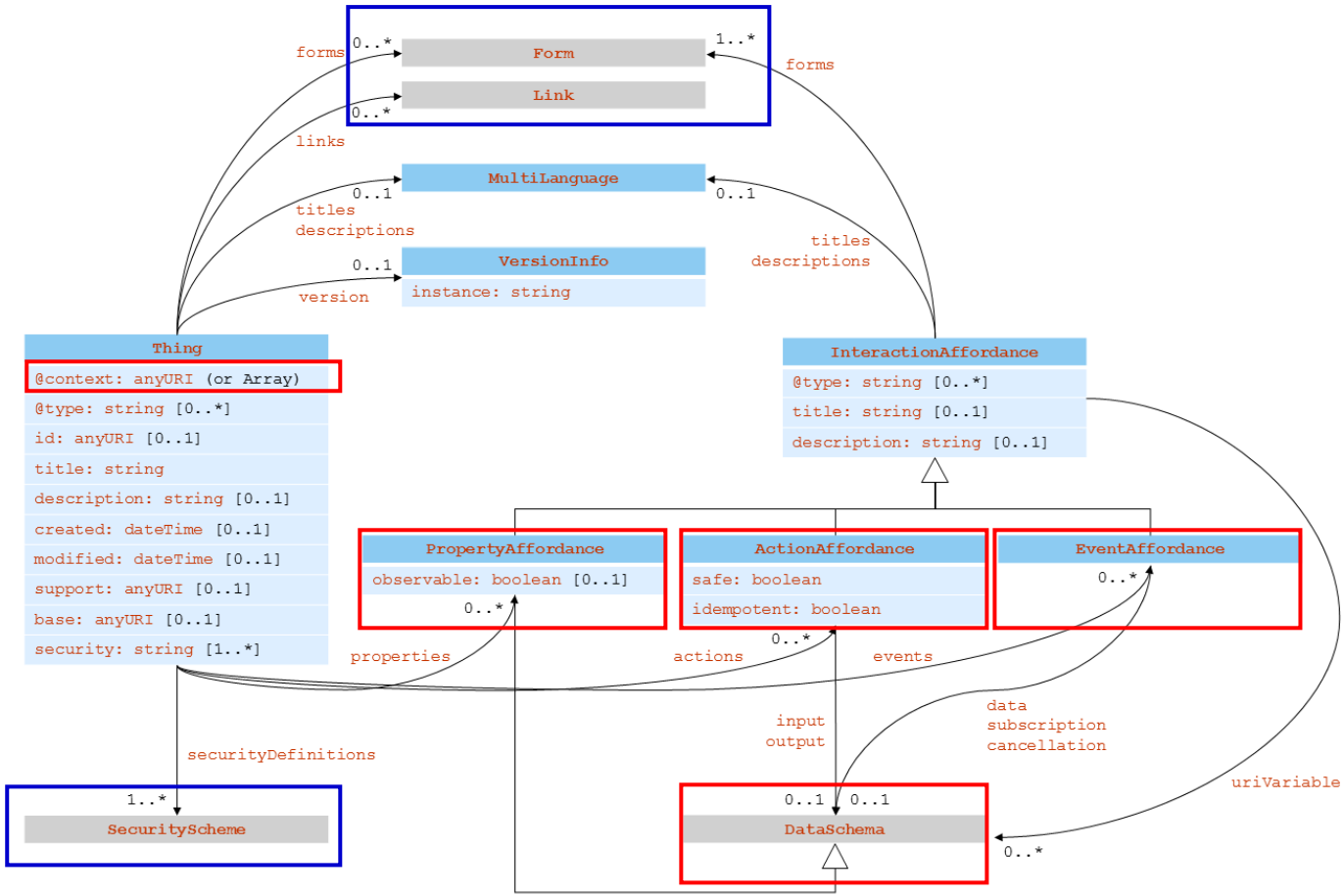
Pattern 3

- WoT Clients talk **directly** with **ELW API** using **information** from **binding templates**

Pattern 1, 2, 3:Use the same **static parts** of **Thing Description**

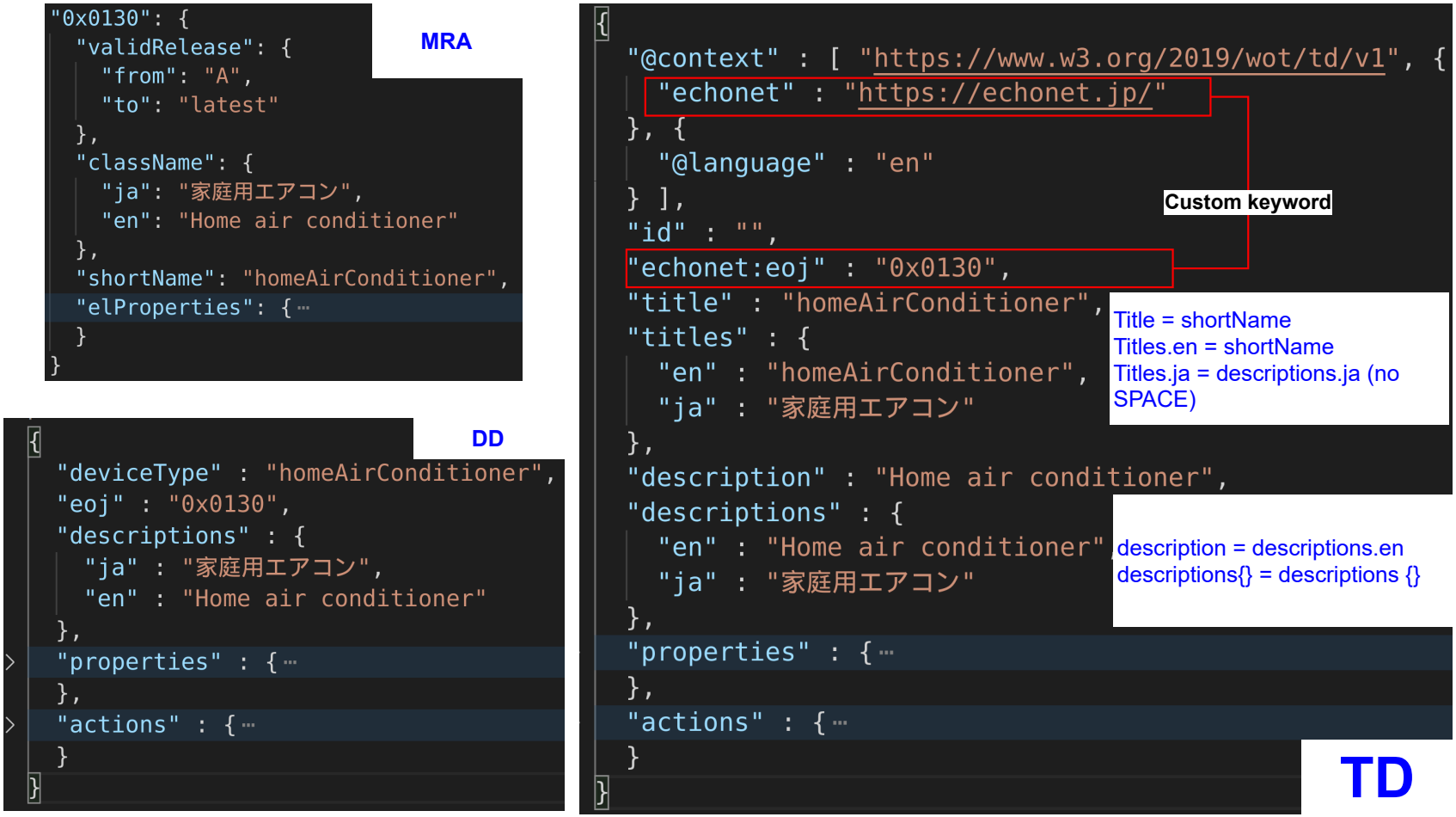


2.1 WoT Thing Description (Overall)



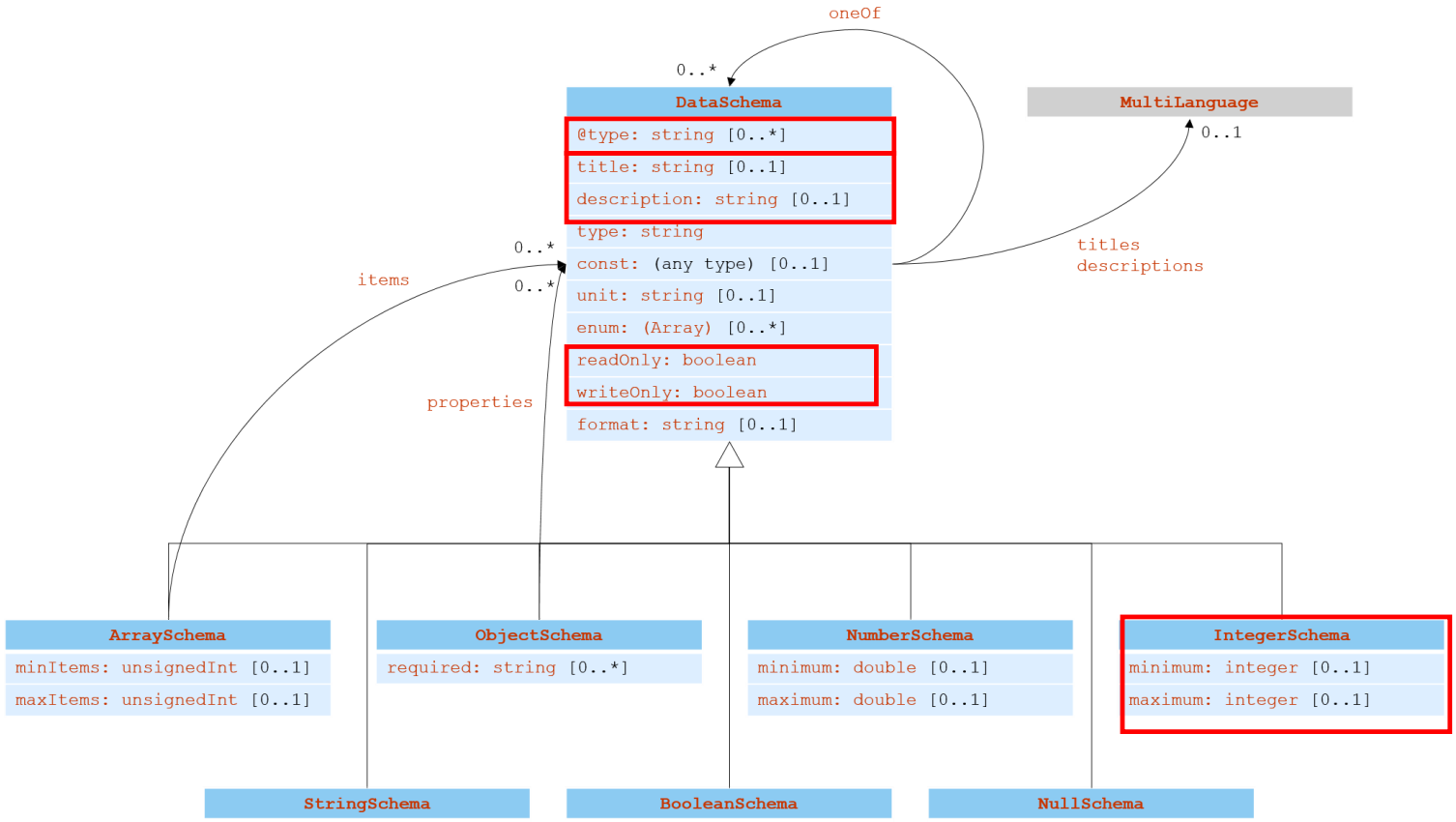
TD. Thing= **DD.**Device, **TD.**Properties= **DD.**Properties, **TD.**Actions= **DD.**Actions, **TD.**Events= **DD.**Events, **TD.**DataSchema= **DD.**Schema

2.1.1 DD/MRA to TD Mapping (Overall)



Customized keywords = (1) define a prefix (e.g *echonet*) $\rightarrow$ (2) **prefix:**propertyName (e.g *echonet:eoj*)

2.2 WoT Thing Description (Data Schema)



TD.@Type= Semantic label, **TD.**title, **TD.**description: in English

TD. IntegerSchema = **DD.**Level ?

→ **Mapping from DD to TD is straightforward**

2.2.1 DD/MRA to TD Mapping (Properties)

setOnly property → **DD.actions** → **TD.**writeOnly is **false** in all cases

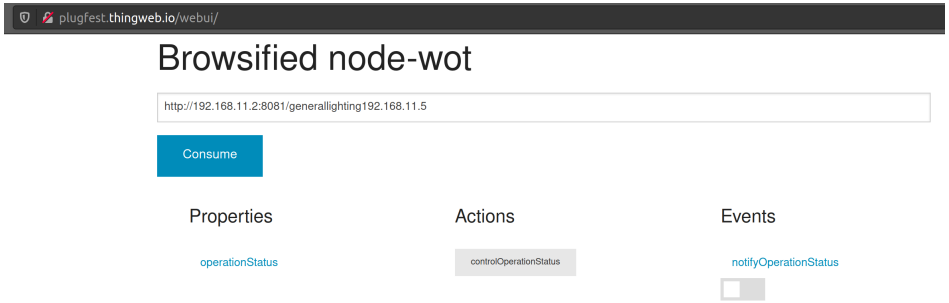
DD.values → **TD.** echonet:edt, value pair ?

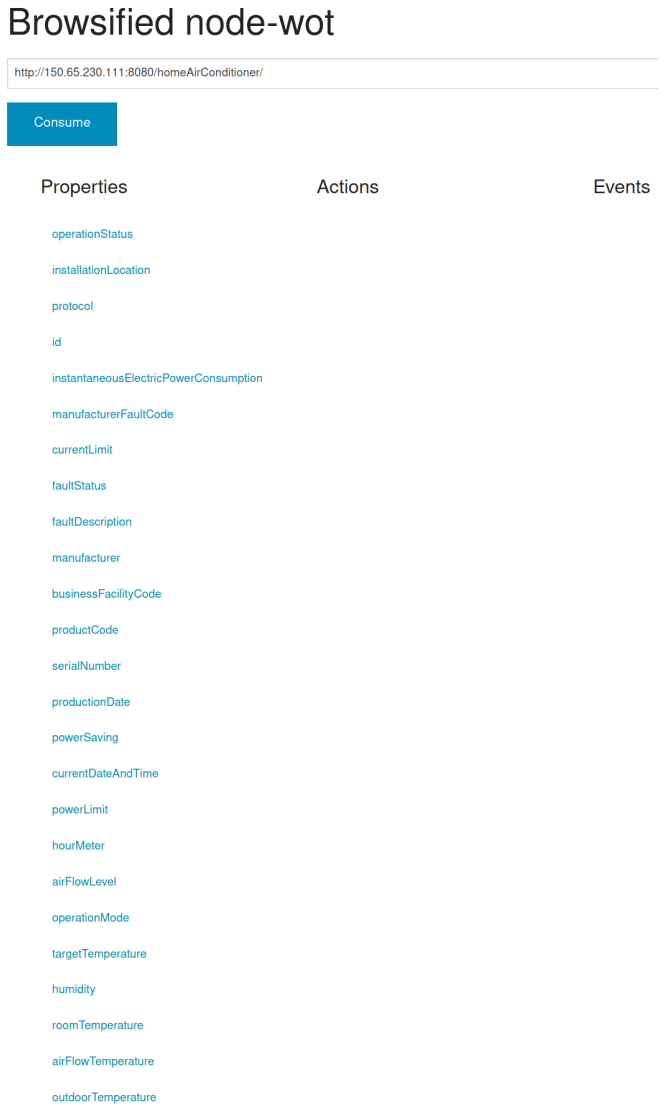
2.2.2 DD/MRA to TD Mapping (Actions/Events)

2.3 Summary

- **Device Description** to **Thing Description** is simply converted using the **rule-based** approach
- **MRA** to **DD and TD** conversion is added to the **eDataModelsGen** tool available at `https://github.com/Tan-Lab/eDataModelsGen`
- All **Thing Descriptions** in this slide are created from MRA automatically using the above tool
- Customized keywords such as **echonet:eoj**, **echonet:epc**, **echonet:edt** are used in **Pattern 1** for **ECHONET Lite Frame** to **TD** conversions
- **TD.** forms, links, securitySchema are available at **run-time** $\rightarrow$ Depend on cooperation patterns

3.1 Expose Thing Description API





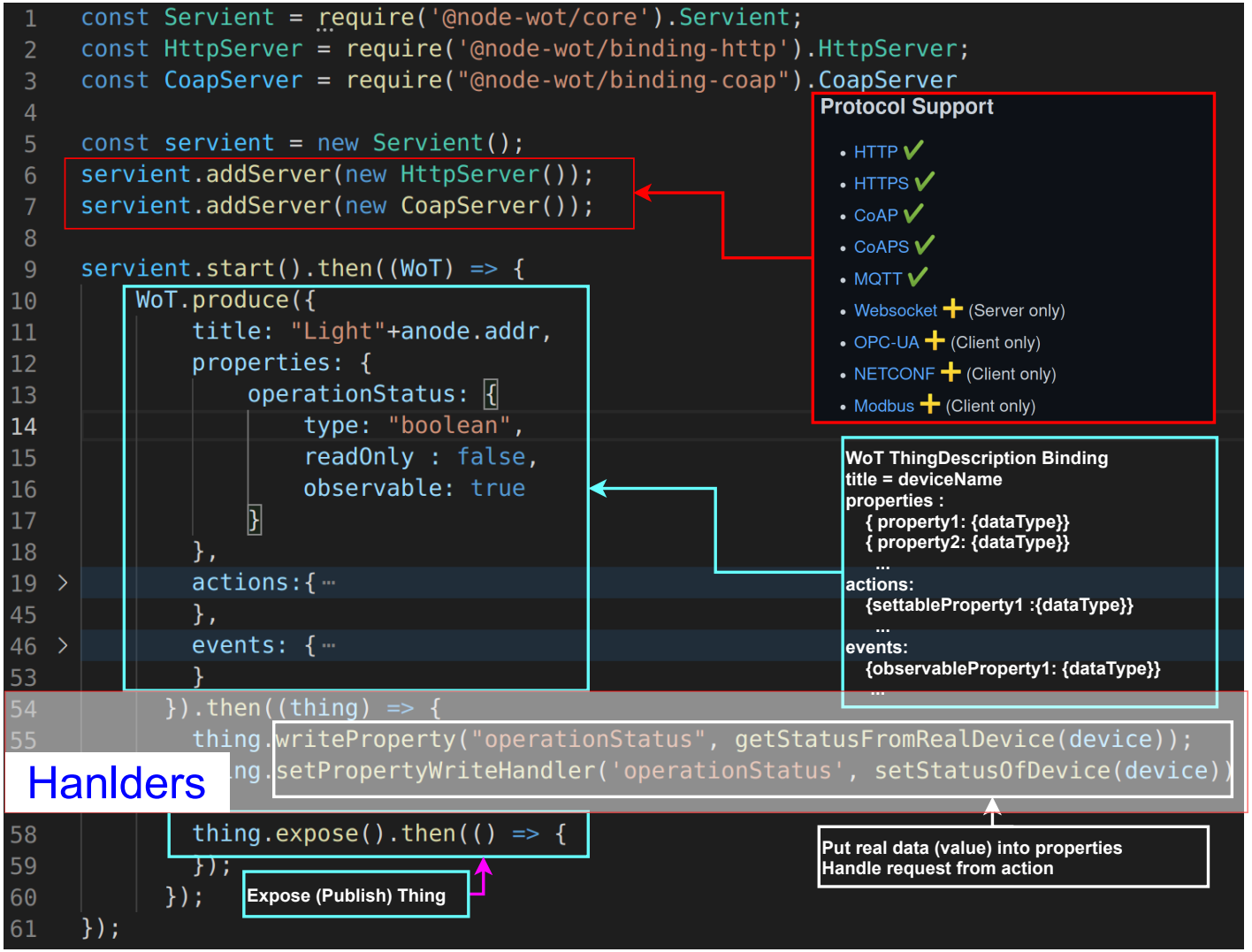
To allow things to be discoverable by WoT clients

- Display **Properties**, **Actions**, **Events**

Library: **node-wot**
`https://github.com/eclipse/thingweb.node-wot`

Sample from
`http://www.thingweb.io/smart-coffee-machine.html`

3.2 Example: Expose a thing by API



Exposing things of Pattern 1, Pattern 2, and **Pattern 3** are **similar**

3.3 Add corresponding handlers

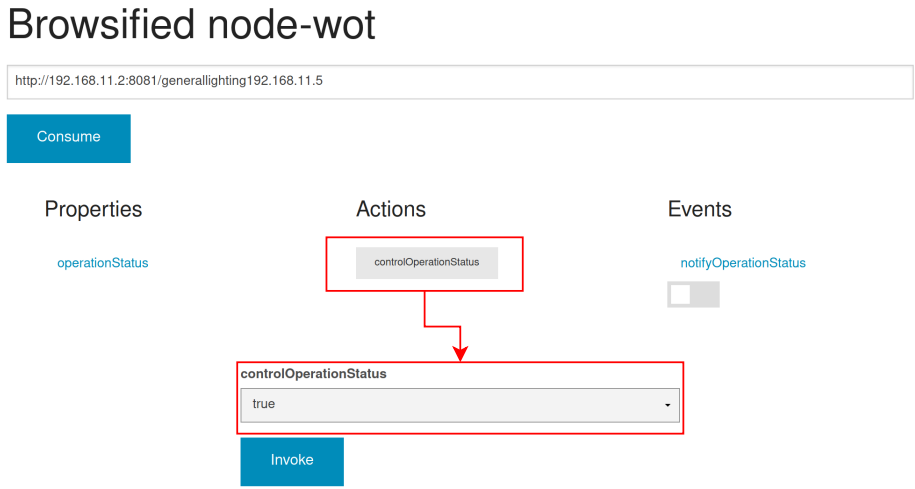
To handle **read, write, invoke action, subcribe events** requests from WoT clients

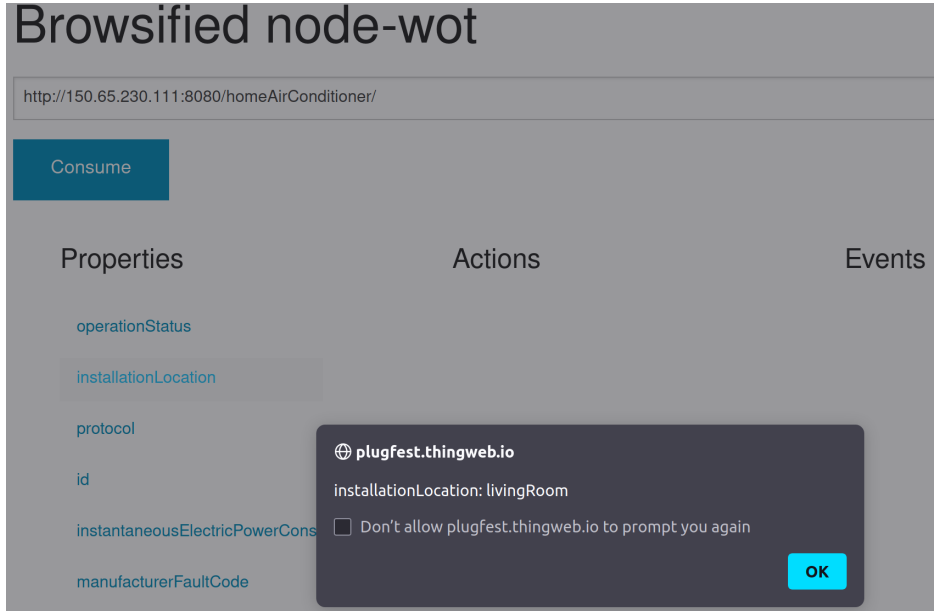
- **not** readOnly → PropertyWriteHandler
- **not** writeOnly → PropertyReadHanlder
- observable or events → emitEvent

  actions → ActionHanlder

Howto:

- **Pattern 1**: send EL frames
- **Pattern 2**: call web APIs
- **Pattern 3**: **No Need handler**





4.1 EL-WoT Proxy Building Blocks

(1) Get ECHONET Lite devices

- **Pattern 1**: Broadcast *NodeFindingMessage* → getInstance list
  - RESULT: **EOJs**
- **Pattern 2**, **Pattern 3**: httpGET /elapi/v1/devices
  - RESULT: **deviceType**, **ID**, etc.

(2) Generate Thing Description

- **Pattern 1**: Compare this.EOJ and TD.echonet:eoj → **xxxTD.json**
- **Pattern 2**, **Pattern 3**: This.deviceType → **xxxTD.json**

→**Not supported properties are included** → **Solution**

- **Pattern 1**: Get **GET, SET, INF** property maps → Gettable, Settable, Observable properties → Compare supported properties and defined property → **xxxTD.json**
- **Pattern 2**, **Pattern 3**: httpGET /elapi/v1/devices/id/properties → Compare supported properties and defined property → **xxxTD.json**

4.1.2 EL-WoT Proxy Building Blocks (cont.)

(3) Expose TD and create **Handlers**

hanlders for Pattern 3 are supported by the ELW API

Manually → Simple, less exception, time-consuming

Automatically → Complex, more exceptions, Fast, **Adaptable**

- **Pattern 1**: Complex dataTypes (objects, array) to Hex codes **problem**

Pattern 2 automatic generated handlers

FOR *property* in *TD.properties*

IF this.property.*is**NOT**WriteOnly* → **PropertyReadHanlder(property)**

httpGET `elapi/v1/devices/{id}/properties/{thisProperty}` → res

IF this.property.*is**NOT**ReadOnly* → **PPWriteHanlder(property, value)**

httpPUT `elapi/v1/devices/{id}/properties/{thisProperty}`, **body:** thisProperty:value → res

4.2 Demonstration with 実験クラウド (Pattern 2)

Pattern 2: Thing Description

→**forms**

- **href**: address of Gateway/Proxy to handle **requests** from WoT Client
- **op**: *readProperty* = **httpGET** (http binding)
- **op**: *writeProperty* = **httpPUT** (http binding)

httpGET, httpPUT requests are handled by **EL-WoT Gateway/Proxy**

→ **Pattern 3**: **href** → ELW API server


```
▼ object {11}
  ▶ @context [3]
    id    : FE00007702905D99427FDB571E1EBDA7D1
    title : FE00007702905D99427FDB571E1EBDA7D1
  ▶ titles {2}
    description : General lighting
  ▶ descriptions {2}
  ▼ properties {21}
    ▼ operationStatus {10}
        title : operationStatus
      ▶ titles {2}
        echonet:epc : 0x80
        description : Operation status
      ▶ descriptions {2}
        readOnly : false
        writeOnly : false
        observable : true
        type : boolean
      ▼ forms [2]
        ▼ 0  {3}
            href : http://192.168.11.2:8080
                   /FE00007702905D99427FDB571E1EBDA7D1/properties
                   /operationStatus
            contentType : application/json
          ▼ op   [2]
              0  : readproperty
              1  : writeproperty
```

4.3 Demonstration with 実験クラウド (Pattern 3)

forms = ELW API + pppath

- propertyPaths are compatible
- **op** read/write are compatible
- **actions** are compatibale
- ✗ writeProperty **body**

  WoT: true/false

  ELW API: operationStatus: true/false
- ✗ getAllProperty **path**

  WoT: *id*/all/properties

  ELW API: *id*/properties


```
▼ object {11}
  ▶ @context [3]
    id    : FE0000770288B93A16E1A5C0C59A16AA32
    title : FE0000770288B93A16E1A5C0C59A16AA32
  ▶ titles {2}
    description : Low-voltage smart electric energy meter
  ▶ descriptions {2}
  ▼ properties {27}
    ▼ operationStatus {10}
        title : operationStatus
      ▶ titles {2}
        echonet:epc : 0x80
        description : Operation status
      ▶ descriptions {2}
        readOnly : false
        writeOnly : false
        observable : true
        type : boolean
      ▼ forms [2]
        ▼ 0  {3}
            href : https://webapiechonet.com/elapi/v1/devices
                   /fe0000770288b93a16e1a5c0c59a16aa32/properties
                   /operationStatus
            contentType : application/json
          ▼ op   [2]
              0  : readproperty
              1  : writeproperty
```

4.4 Solutions for Pattern 3

- WoT Generated **href** = my **href** → not yet supported by current **node-wot** library → Pull Request?

```
// set contentType (extend with more?)
public static updatePropertyFormWithTemplate(form: TD.Form, tdTemplate: WoT.ThingDescription, propertyName: string) {
  if (form && tdTemplate && tdTemplate.properties && tdTemplate.properties[propertyName] && tdTemplate.properties[propertyName].forms) {
    for (let formTemplate of tdTemplate.properties[propertyName].forms) {
      // 1. Try to find match with correct href scheme
      if (formTemplate.href) {
        // TODO match for example http only? form.href = formTemplate.href
      }

      // 2. Use any form
      if (formTemplate.contentType) {
        form.contentType = formTemplate.contentType;
        return; // abort loop
      }
    }
  }
}
```

- ELW API httpPUT **body** → **only data** → remove propertyName from body

5 Summary

- WoT Binding Templates for ECHONET Lite = EL-WoT **Thing Description** + **Protocol Bindings**
- Thing Description Schemes are generated from Device Descriptions
  - Auto conversion tool is available (**eDataModelsGen**)
- Three Patterns for EL-WoT Cooperation are implemented
- Mechanism to create WoT-ECHONET Lite Gateway
  - Init WoT Servient
  - Init ECHONET Lite interface → When detect a device → create Thing Description for the device + getValue from device → expose device as WoT consumable resources
  - **Pattern 1 and 2** →Create operation handlers for operations
  - **Pattern 3** → Redirect to corresponding ELW API
- Need some adjustments of ELW API to implement **Pattern 3**

  EL-WoT Proxy, **EL-WoT Binding Proxy**:
  `https://github.com/Cupham/eWoTProxy`

  MRA-DD-TD Conversion tool:`https://github.com/Tan-Lab/eDataModelsGen`