

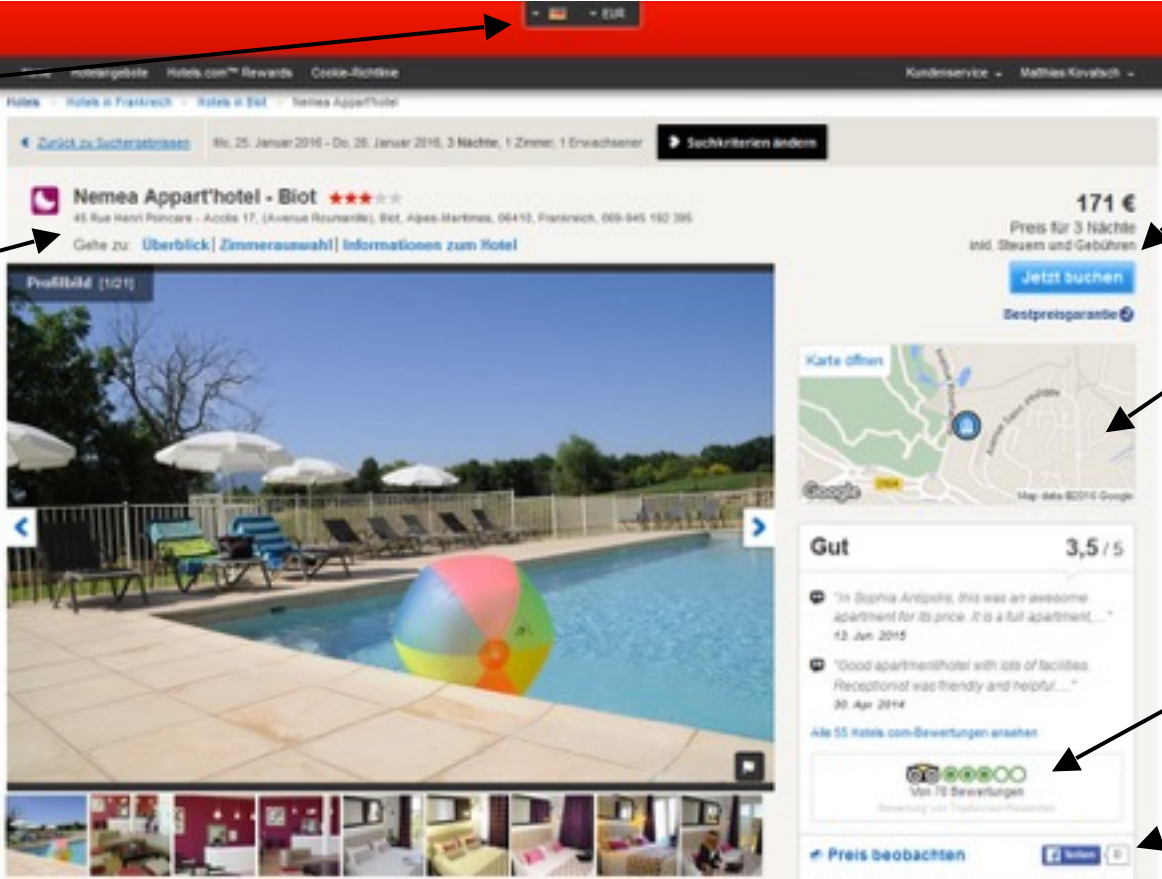
Overview of HATEOAS Approaches

W3C WoT IG / IRTF T2TRG Joint Meeting, Nice, France, 2016

Matthias Kovatsch (kovatsch@inf.ethz.ch)

Web Mashups through Open APIs

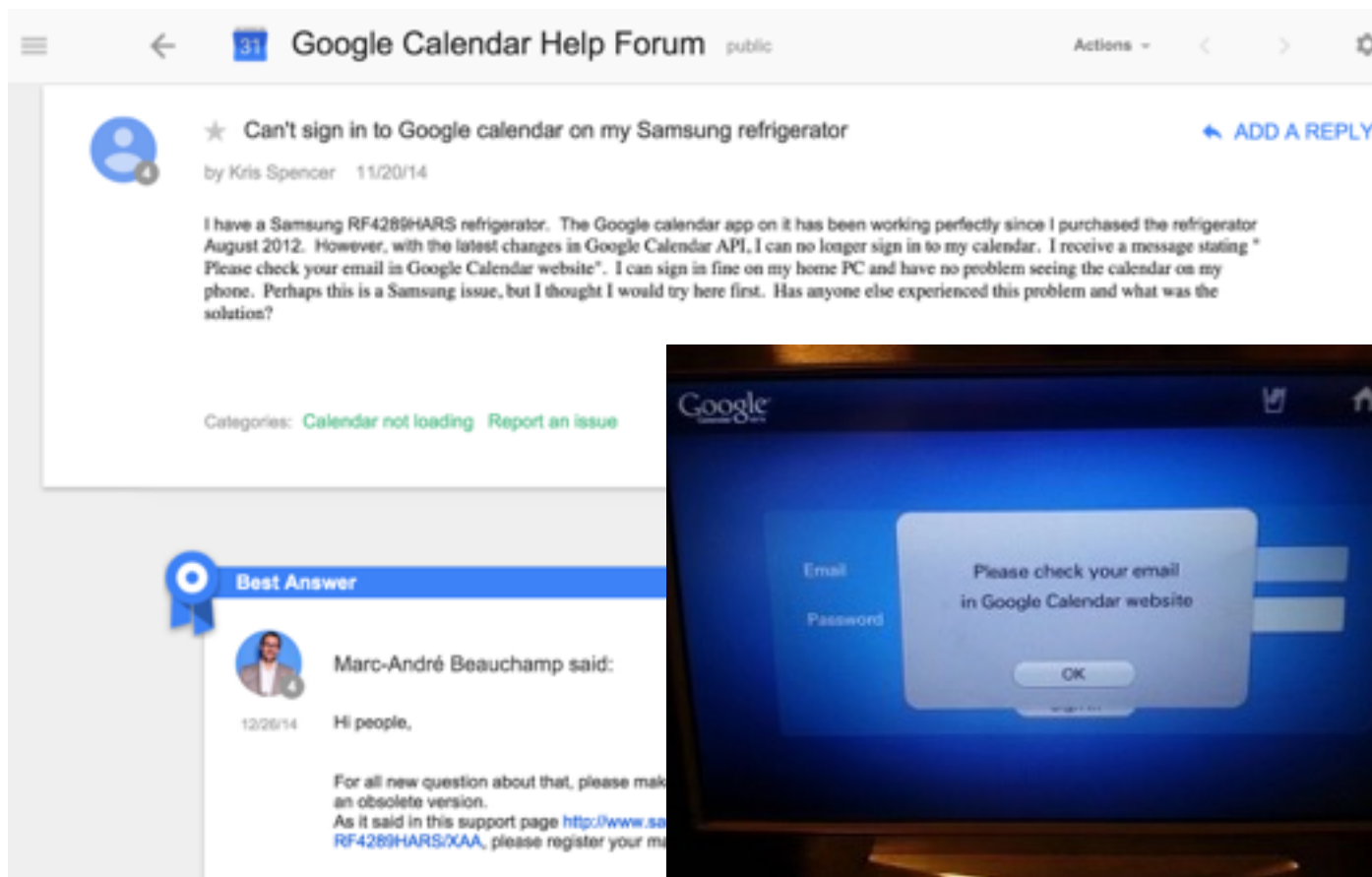
Internal microservice APIs



tripadvisor®

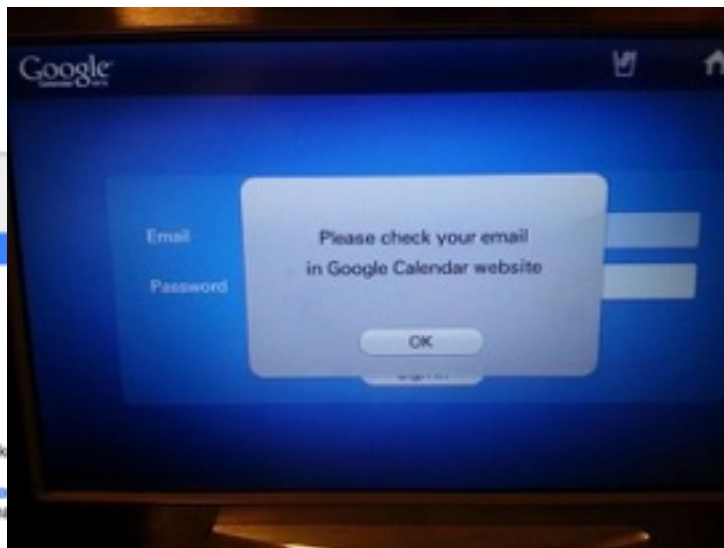


... Often Break

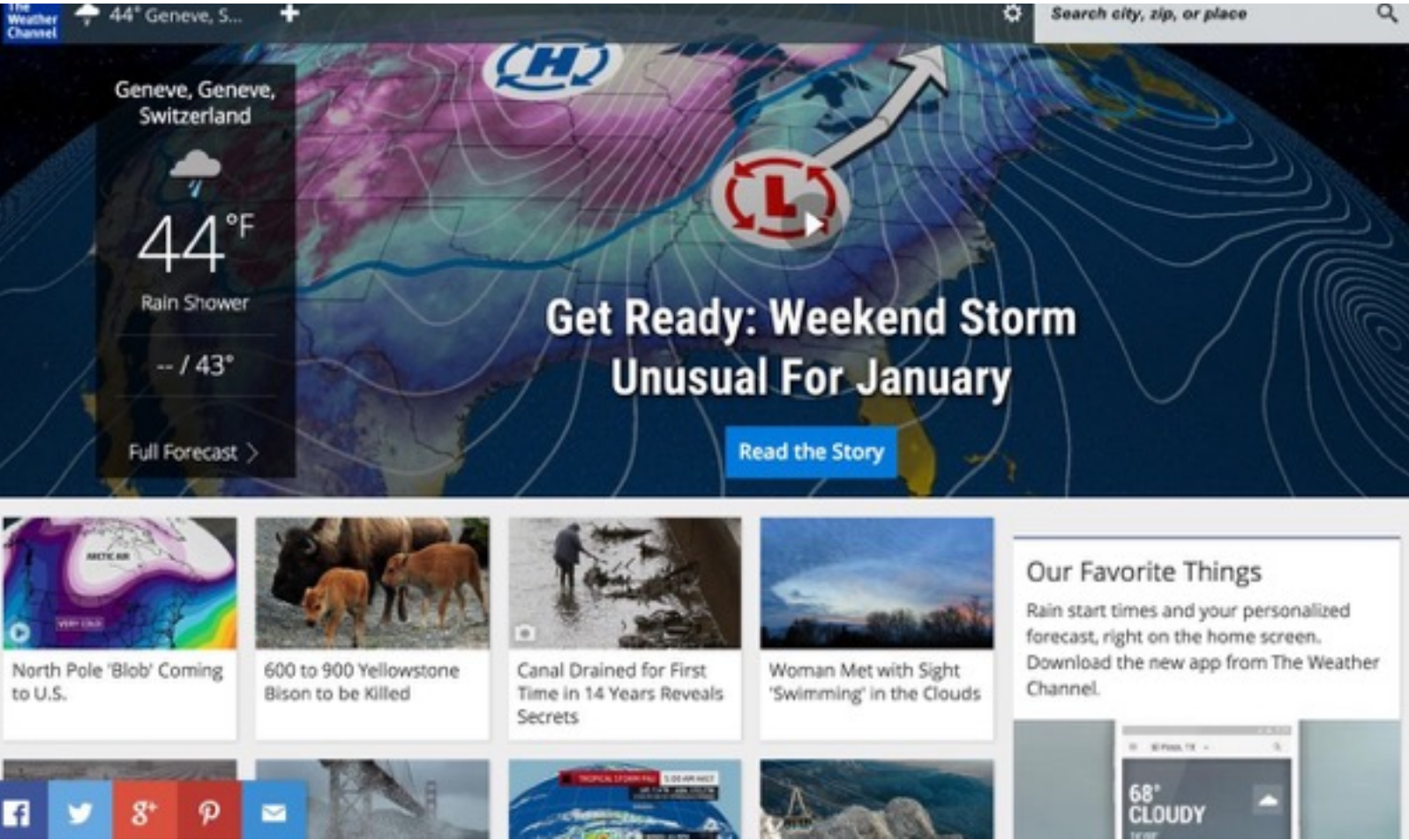


The screenshot shows a forum post titled "Can't sign in to Google calendar on my Samsung refrigerator" by Kris Spencer, dated 11/20/14. The post describes a problem with the Google Calendar app on a Samsung RF4289HARS refrigerator. The user reports that the app worked perfectly since purchase in August 2012 but now fails to sign in due to API changes, displaying an error message: "Please check your email in Google Calendar website". The user asks if this is a Samsung issue and if anyone else has experienced this problem.

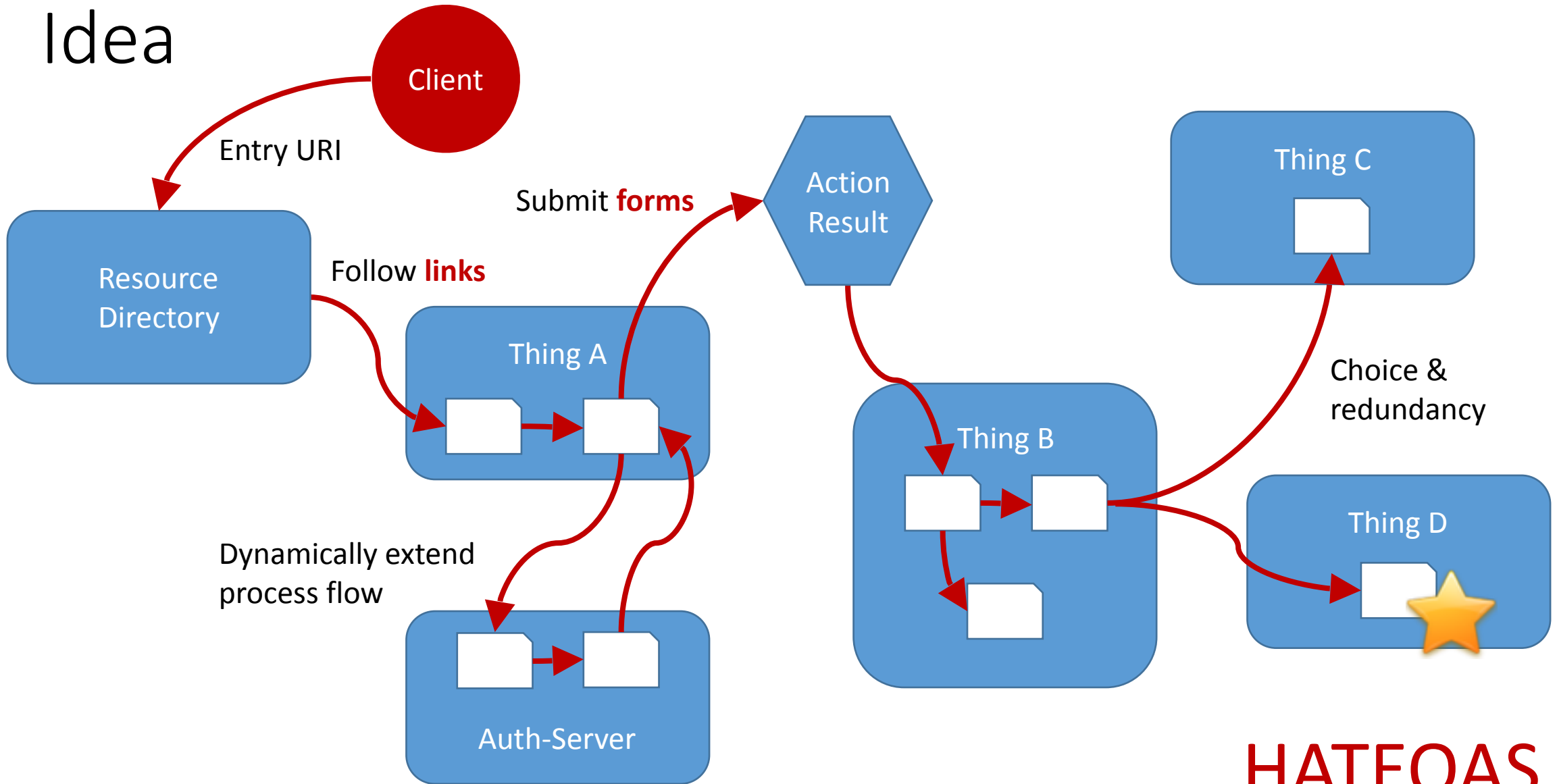
Below the post, a "Best Answer" is provided by Marc-André Beauchamp, dated 12/26/14. He advises that for new questions, users should use an obsolete version of the app and refer to a support page: <http://www.samsung.com/boards/question/2014-11-20/RF4289HARS/XAA>.



Human Web Interaction



Idea



HATEOAS

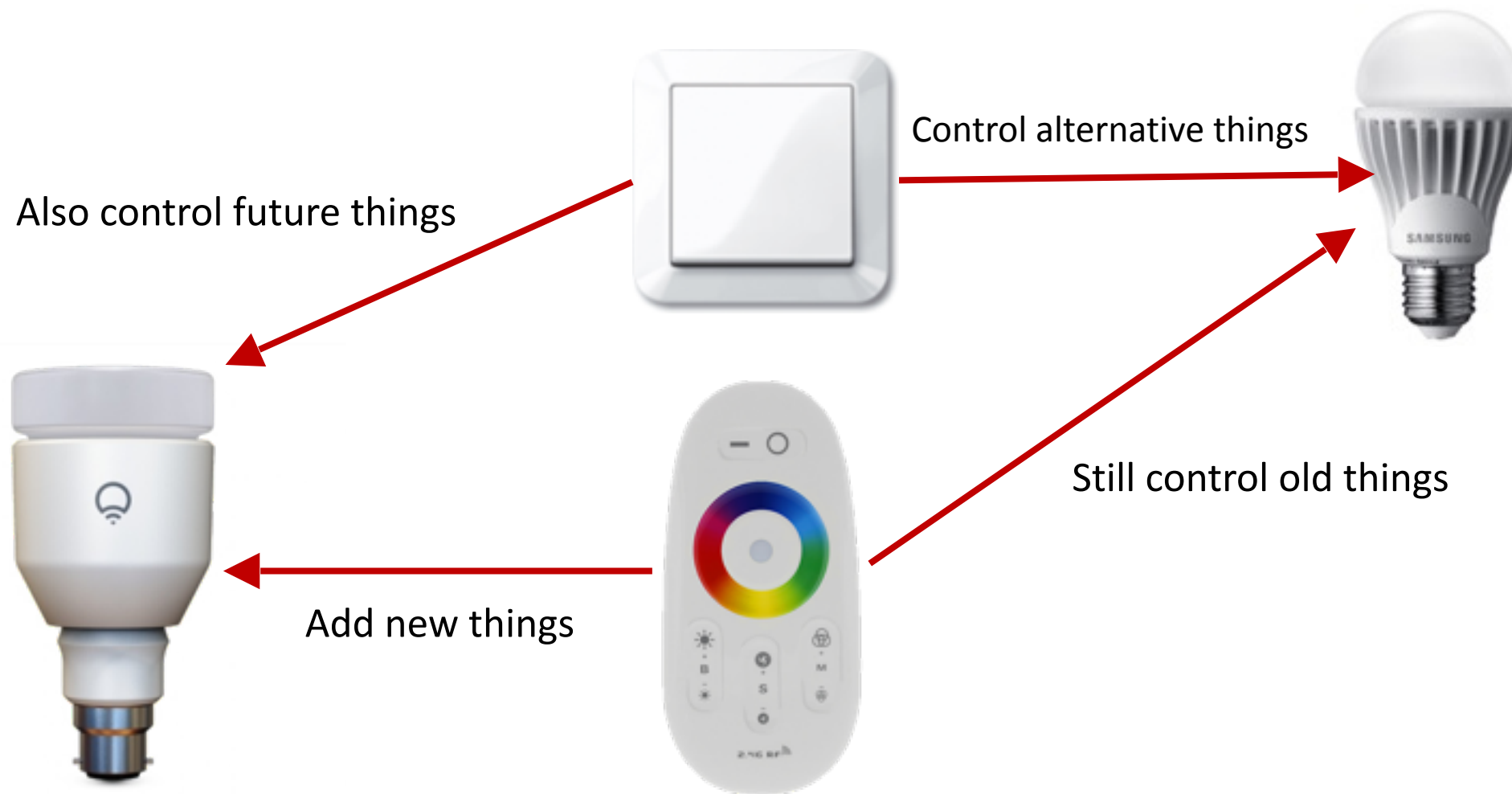
HATEOAS Summary

- **Function descriptions** are in-band (optimizations always an option)
- Clients can learn applications on the fly
 - Links and forms define control flow (functional descriptions)
 - Relation types provide semantic annotations
- Servers are free to define their own resource structure
 - Structure in multiple components
 - Dynamically add new or proprietary features
- **Loose Coupling**
 - Clients and servers can evolve independently
 - Clients can adapt to changing environments

Practical Approach: PlugREST

- Starting point: Lighting use case
 - Well-known requirements
 - Many non-REST implementations (ZigBee etc.)
- Plan
 - Define basic components (lights and controllers)
 - Evolve over time by adding features
 - Connect to smart energy domain for cross-domain evolution
 - Learn and test how to optimally **handle change** through REST
 - Learn more about representation format design
 - Learn more about machine-understandable hypermedia controls
 - Test independent client and server evolution

Change



CoRE Apps and CoRE Lighting

Individual drafts by Klaus Hartke

hartke@tzi.org

Hypertext-driven Applications

- CoRE Application Descriptions

<https://tools.ietf.org/html/draft-hartke-core-apps-02>

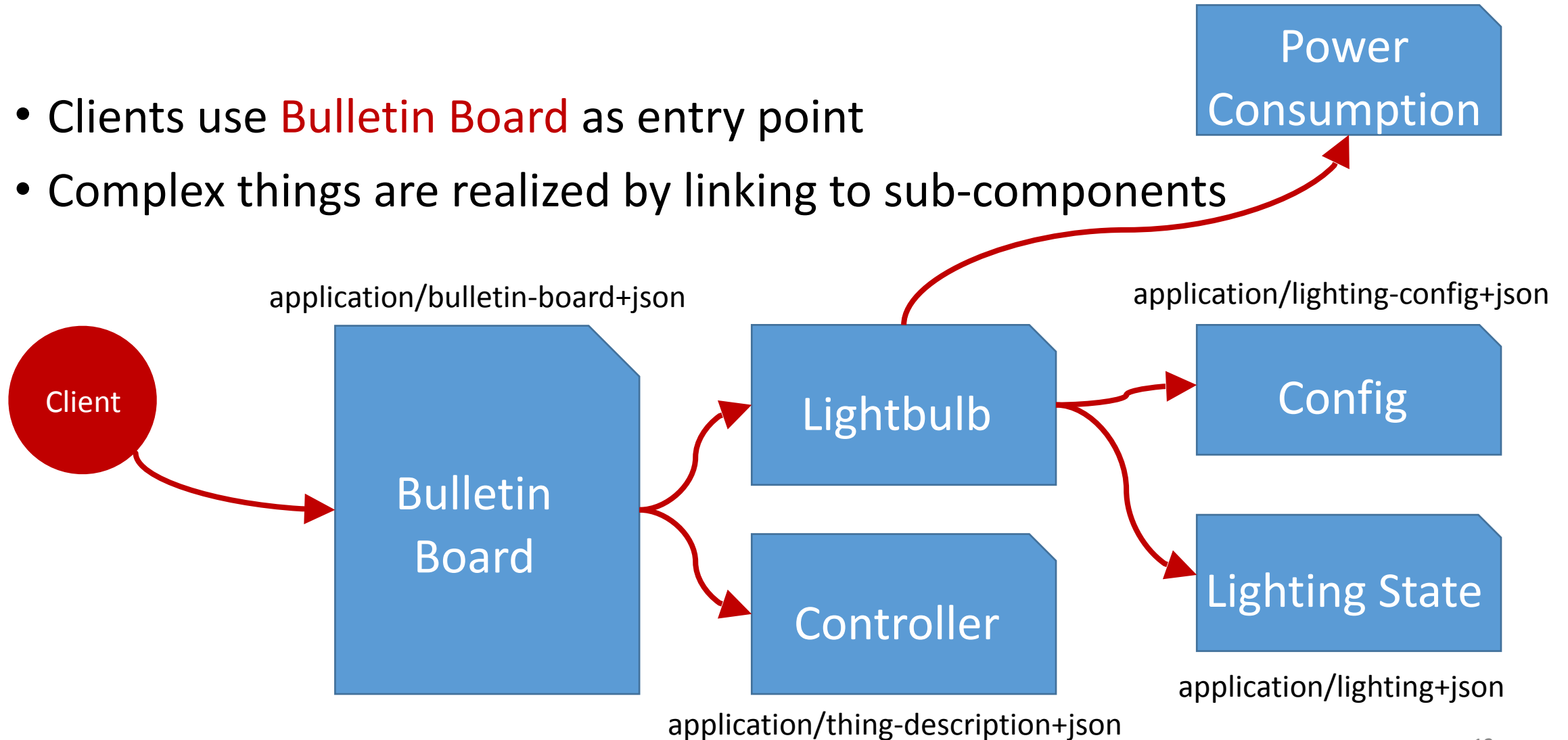
- Describe REST applications fully by
 - **URI schemes** that identify communication protocols,
 - **Internet Media Types** that identify representation formats,
 - **link relation types** that identify link semantics,
 - **form relation types** that identify form semantics, and
 - optionally, **well-known locations** as entry points

draft-hartke-core-lighting-00

- Scenario and component description
 - Light bulbs
 - Light Remote Controls (LRCs)
- Application-specific Internet Media Types
 - **Give meaning to JSON properties**
 - application/bulletin-board+json (**discovery**)
 - application/thing-description+json (**thing metadata**)
 - application/lighting-config+json (**binding**)
 - application/lighting+json (**lighting state control**)
 - **Hypermedia controls** based on HAL (Hypertext Application Language)
 - No templated links or CURIEs
 - Enriched with form support

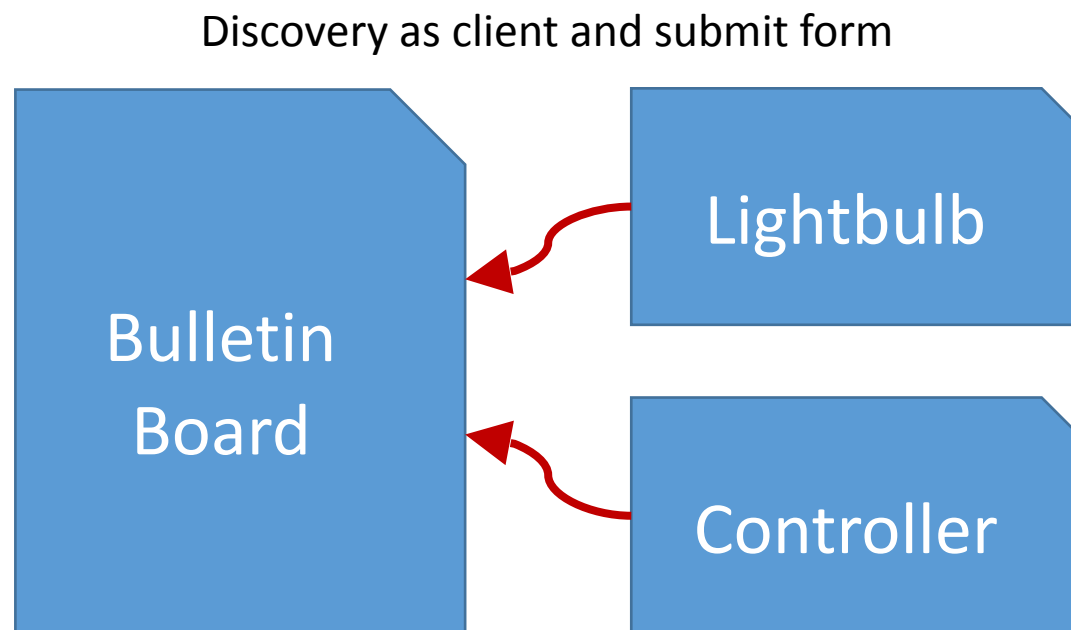
Discovery through Links

- Clients use **Bulletin Board** as entry point
- Complex things are realized by linking to sub-components



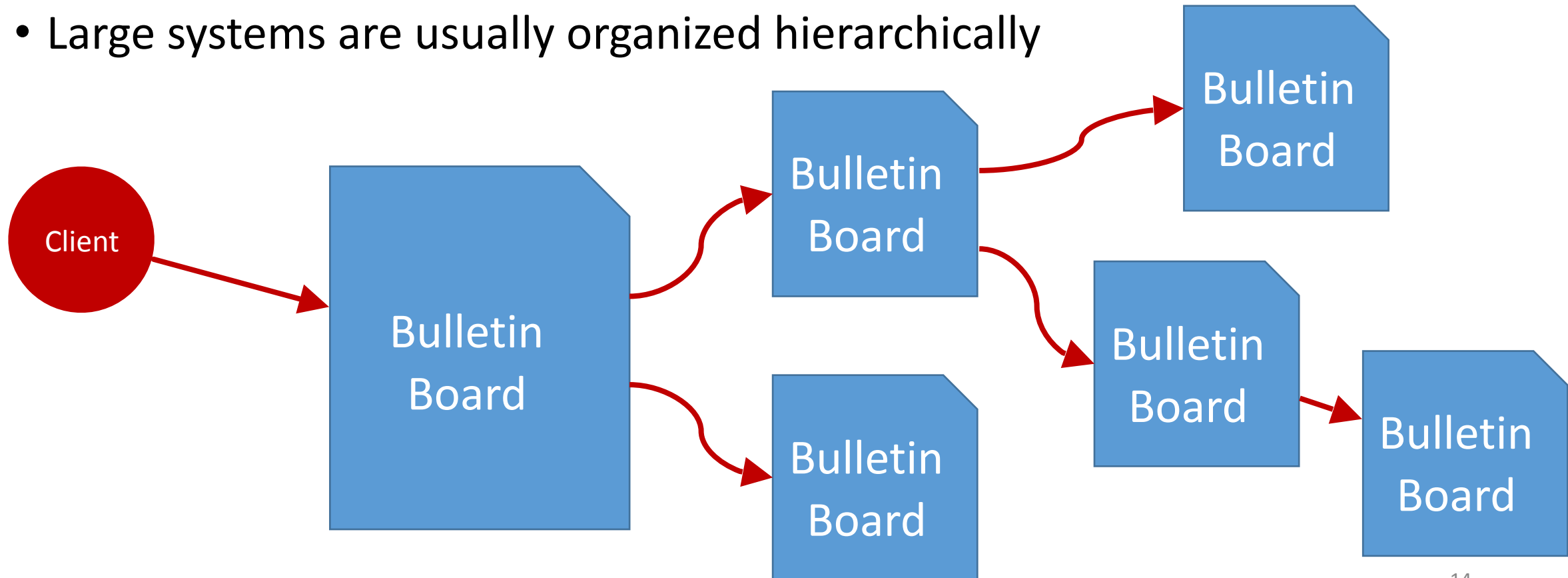
Discovery through Links

- Things register with the Bulletin Board
- application/bulletin-board+json provides a «**create item**» form



Discovery through Links

- Linking during discovery allows distribution and delegation (cf. DNS)
- Large systems are usually organized hierarchically



Links

- <https://tools.ietf.org/html/draft-hartke-core-apps-02>
- <https://tools.ietf.org/html/draft-hartke-core-lighting-00>

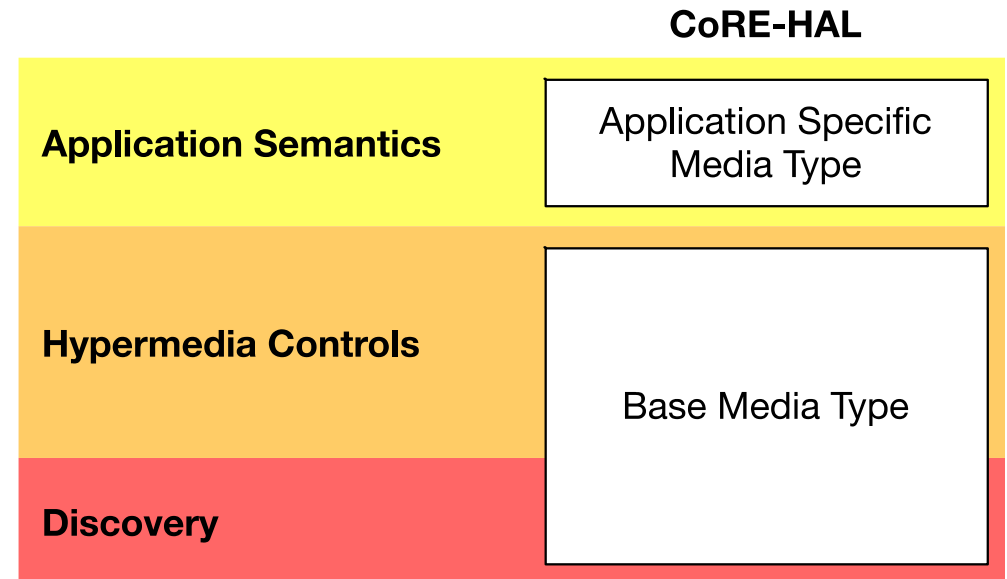
CoRE-HAL and Hypermedia Client

Extensions by Matthias Kovatsch and Yassin Hassan

kovatsch@inf.ethz.ch

Split Internet Media Type Definitions

- CoRE-HAL base media type
 - For now JSON (without -LD)
 - Hypermedia controls (links and forms)
 - Thing Description properties
 - Location Description properties
- Application-specific media types
 - Per atomic use case
 - Data model
 - Link and form relation types
 - Group semantic vocabulary



CoRE-HAL Lighting State Example

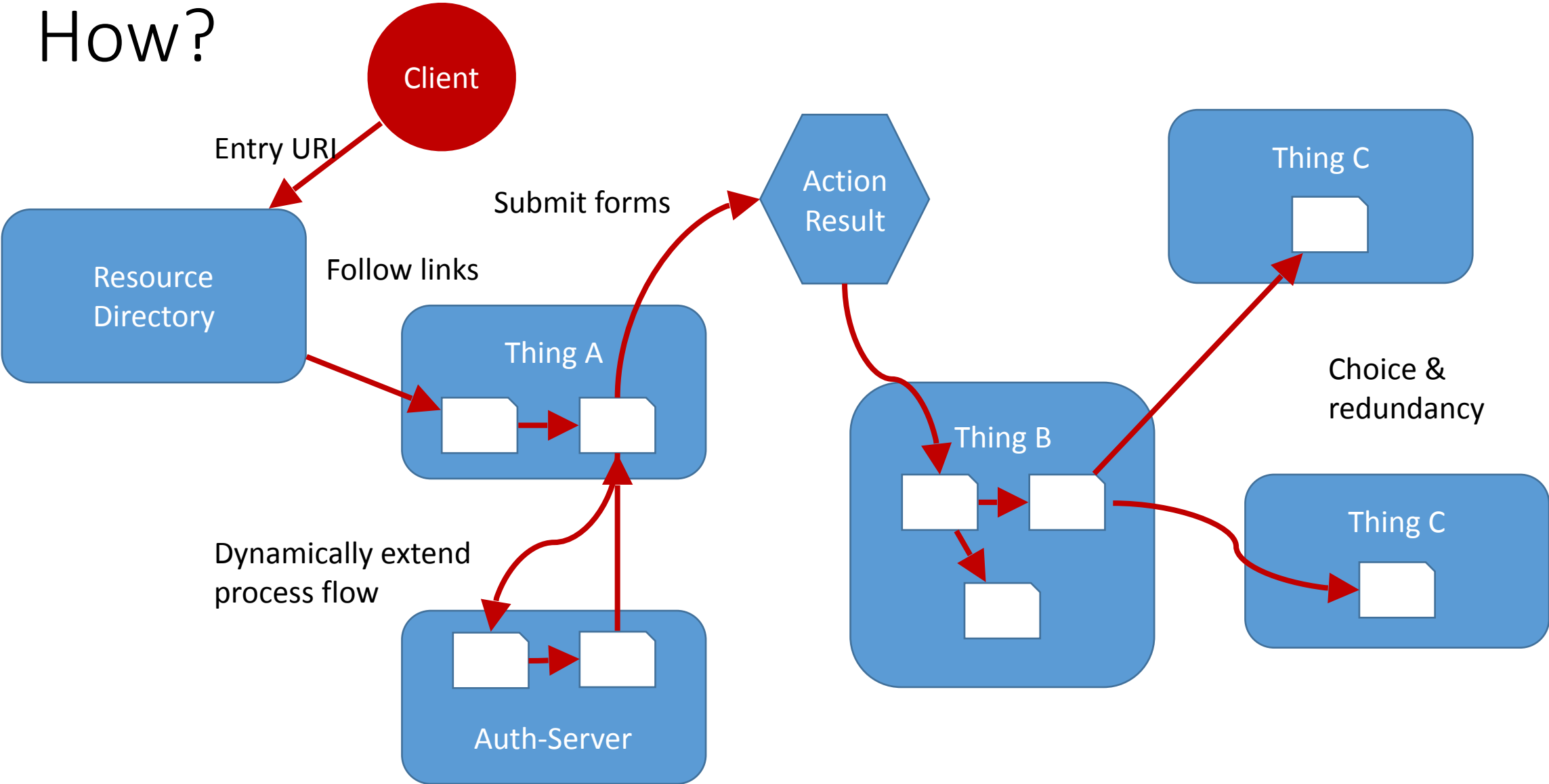
```
{
  "value": {"r":255, "g":0, "b":0},
  "mode": "rgb",
  "_links": {
    "same-as": {
      "href": "/brightness",
      "type": "application/x.lighting-state+json"
    }
  },
  "_forms": {
    "edit": {
      "method": "PUT",
      "href": "/light",
      "accepts": "application/x.lighting-state+json"
    }
  },
  "_self": "/light"
}
```

Application Data

Links

Forms

How?

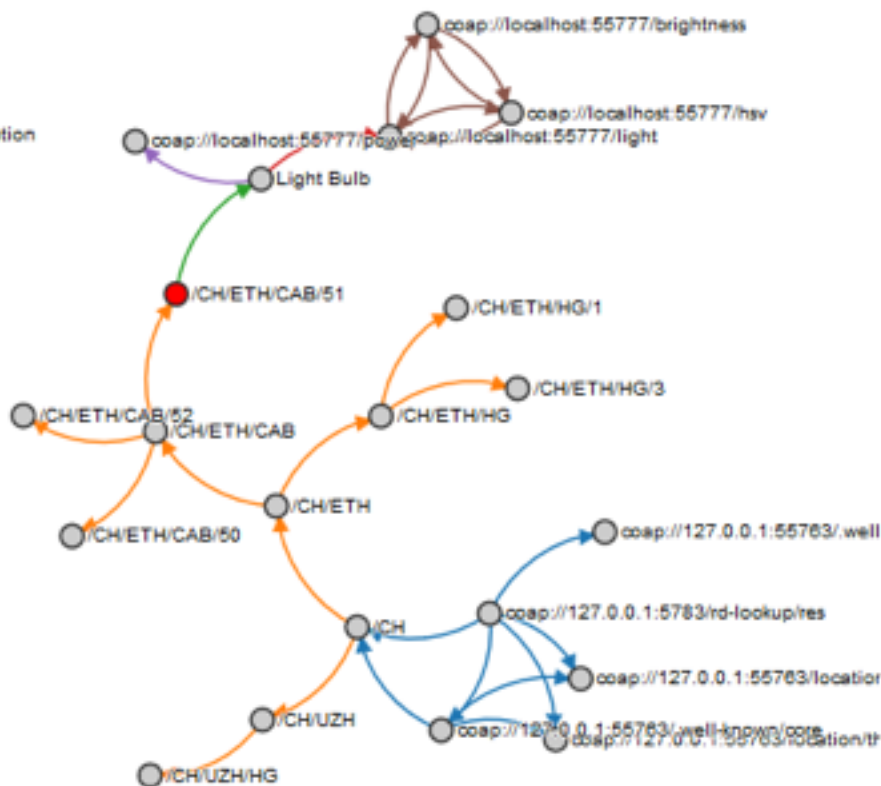


CoRE-HAL Explorer

coap://127.0.0.1:5783/rd-lookup/res

Run!

- initial
- child
- thing
- lighting-state
- power-consumption
- same-as



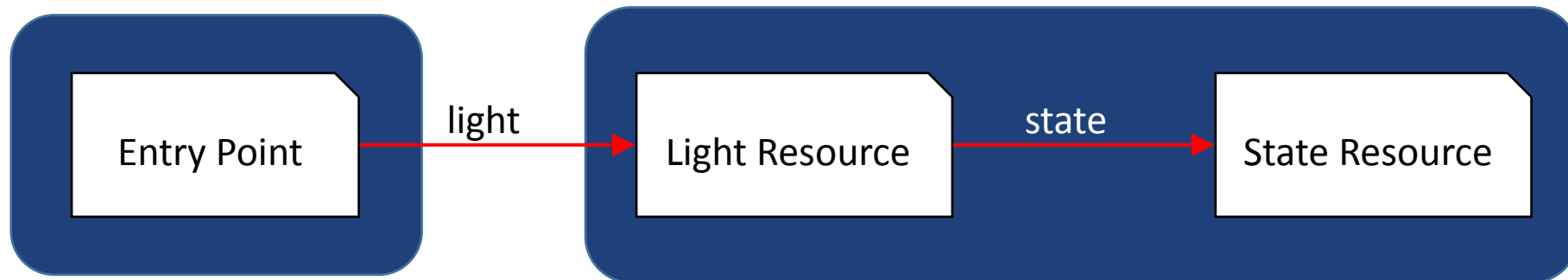
coap://localhost:55768/location

```
{
  location: "/CH/ETH/CAB/51",
  -_links: {
    - thing: {
      href: "coap://localhost:55775/thing",
      type: "application/x.thing-description+json"
    }
  },
  -_forms: {
    - remove-thing: {
      href: "/location/things(?_self)",
      method: "DELETE",
      accepts: "application/x.thing-description+json",
      templated: true
    },
    - add-child: {
      href: "/location/children",
      method: "POST",
      accepts: "application/x.location-description+json"
    },
    - add-thing: {
      href: "/location/things",
      method: "POST",
      accepts: "application/x.thing-description+json"
    },
    - edit: {
      href: "",
      method: "POST",
      accepts: "application/x.location-description+json"
    }
  }
},
```

Hypermedia Client

- High-level path description to resource based on link relation types
- Actual (dynamic) URIs are retrieved from representations

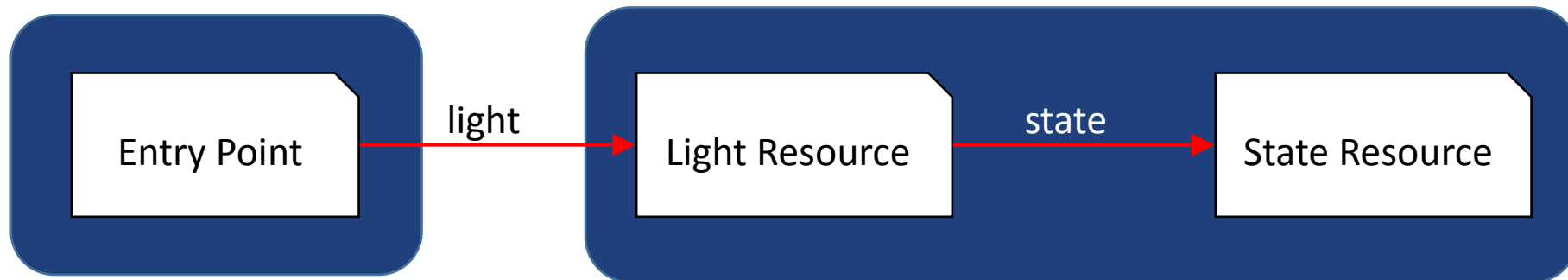
```
ep    = new HypermediaClient("coap://home.local"); // entry point  
light = ep.followLink("light"); // link relation type  
state = light.followLink("state"); // link relation type
```



Hypermedia Client Futures

- Lazy loading of resource representations
- Only request representations (i.e., transmit data) when used

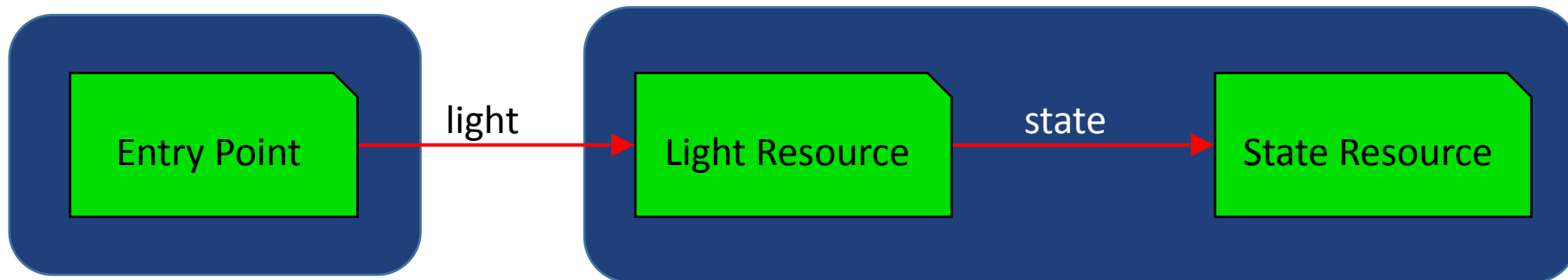
```
ep    = new HypermediaClient("coap://home.local");  
light = ep.followLink("light");  
state = light.followLink("state");  
data  = state.get();
```



Hypermedia Client Futures

- Reloadable resource representation in the future
- Transparently handles cache control

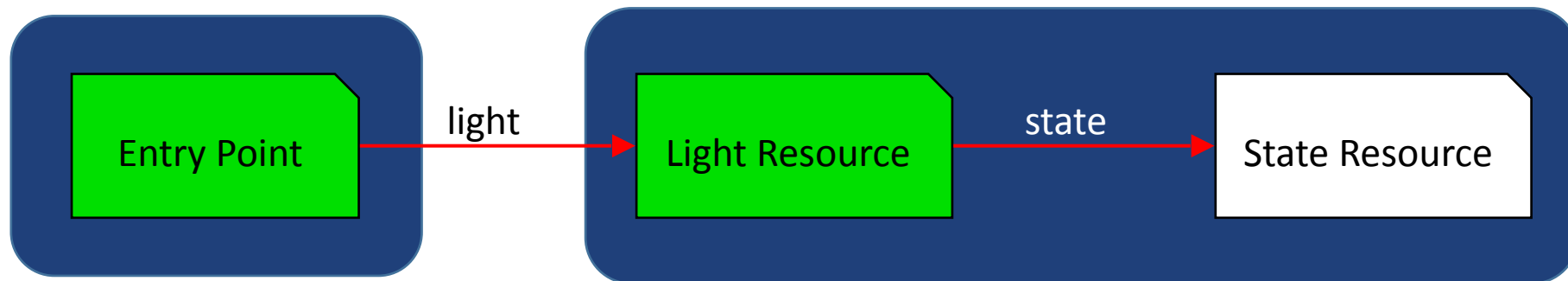
```
ep    = new HypermediaClient("coap://home.local");  
light = ep.followLink("light");  
state = light.followLink("state");  
data  = state.get();  
// Max-Age expires ...  
data  = state.get();
```



Hypermedia Client Futures

- Bookmark support
- On error discovery is re-triggered to recover from unavailable/replaced devices

```
// thing is replaced, address and resource path changes  
data = state.get();
```

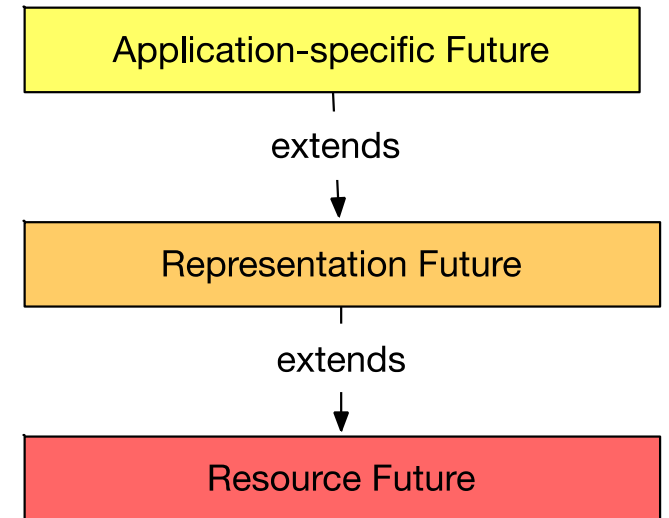


Media-type-specific Future Object

- Programmatically provide application-specific operations
- Allow developer to use the IDE auto-completion feature

```
public class LightingStateFuture
    extends CoREHalResourceFuture<LightingState> {

    public void setRGB(int r, int g, int b) {
        LightingState lightingState = new LightingState();
        lightingState.setValue(new RGBValue(r,g,b));
        submitForm("edit", lightingState);
    }
}
```



Links

- <http://ynh.github.io/core-hal-explorer/>
- <https://github.com/ynh/coap-polyfill>
- Java Hypermedia Client and Actinium JavaScript module will be released publicly soon, announcement on T2TRG mailing list (<https://www.irtf.org/mailman/listinfo/t2trg>)

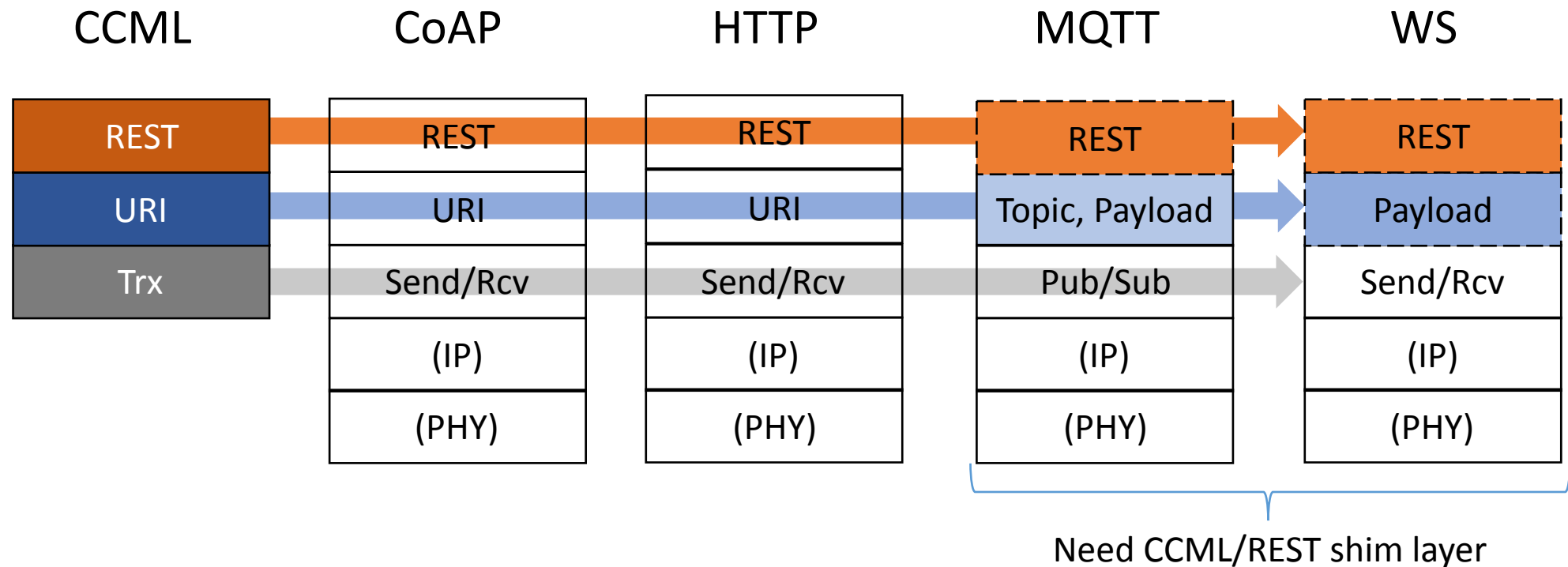
Hypermedia System Architecture

Approach by Michael Koster

michael.koster@smarthings.com

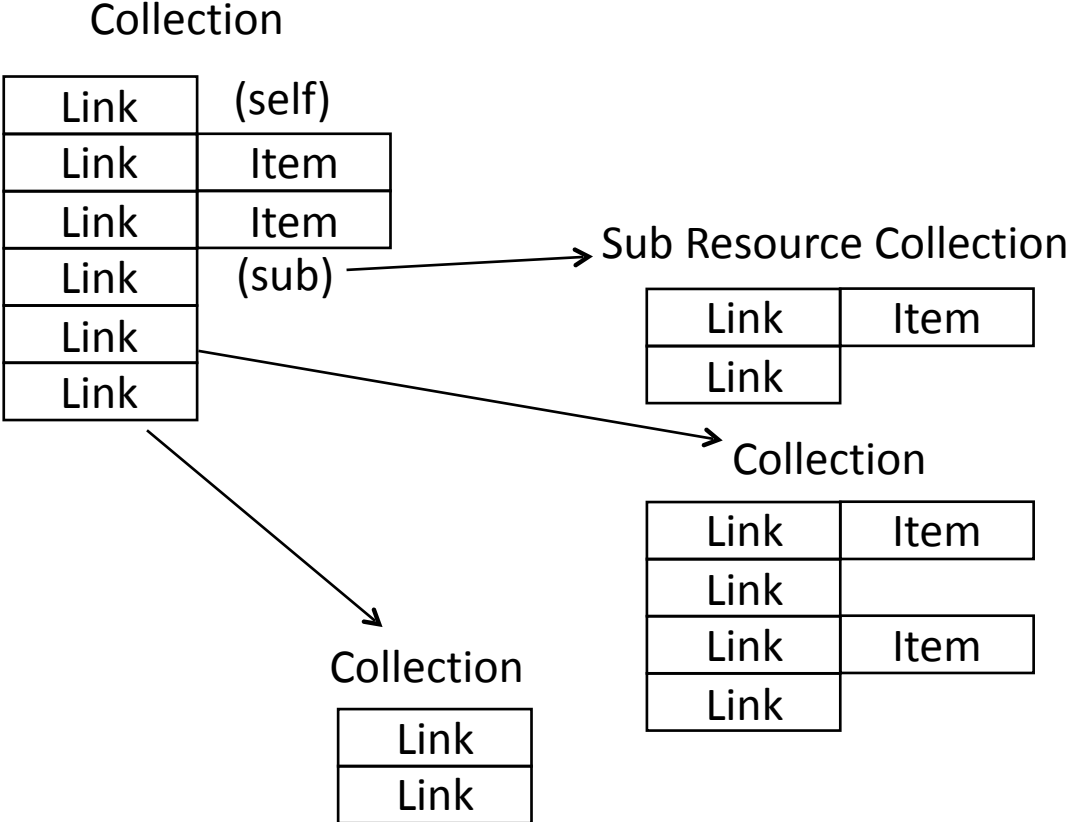
Common CRUD Model for Protocol Bindings

- Map abstraction to HTTP and CoAP request and responses
- Encapsulate the abstraction in WS and MQTT payloads

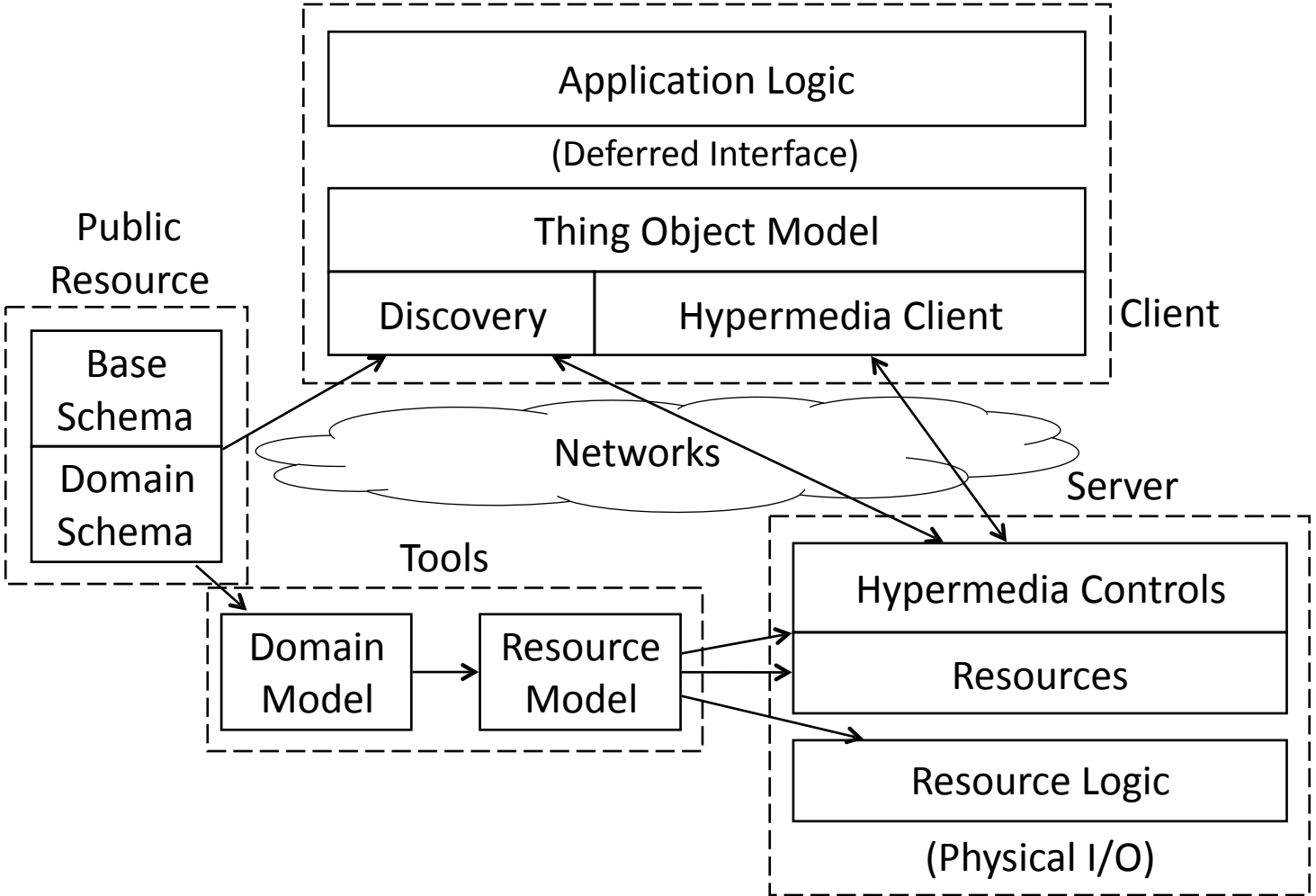


Internet Media Types

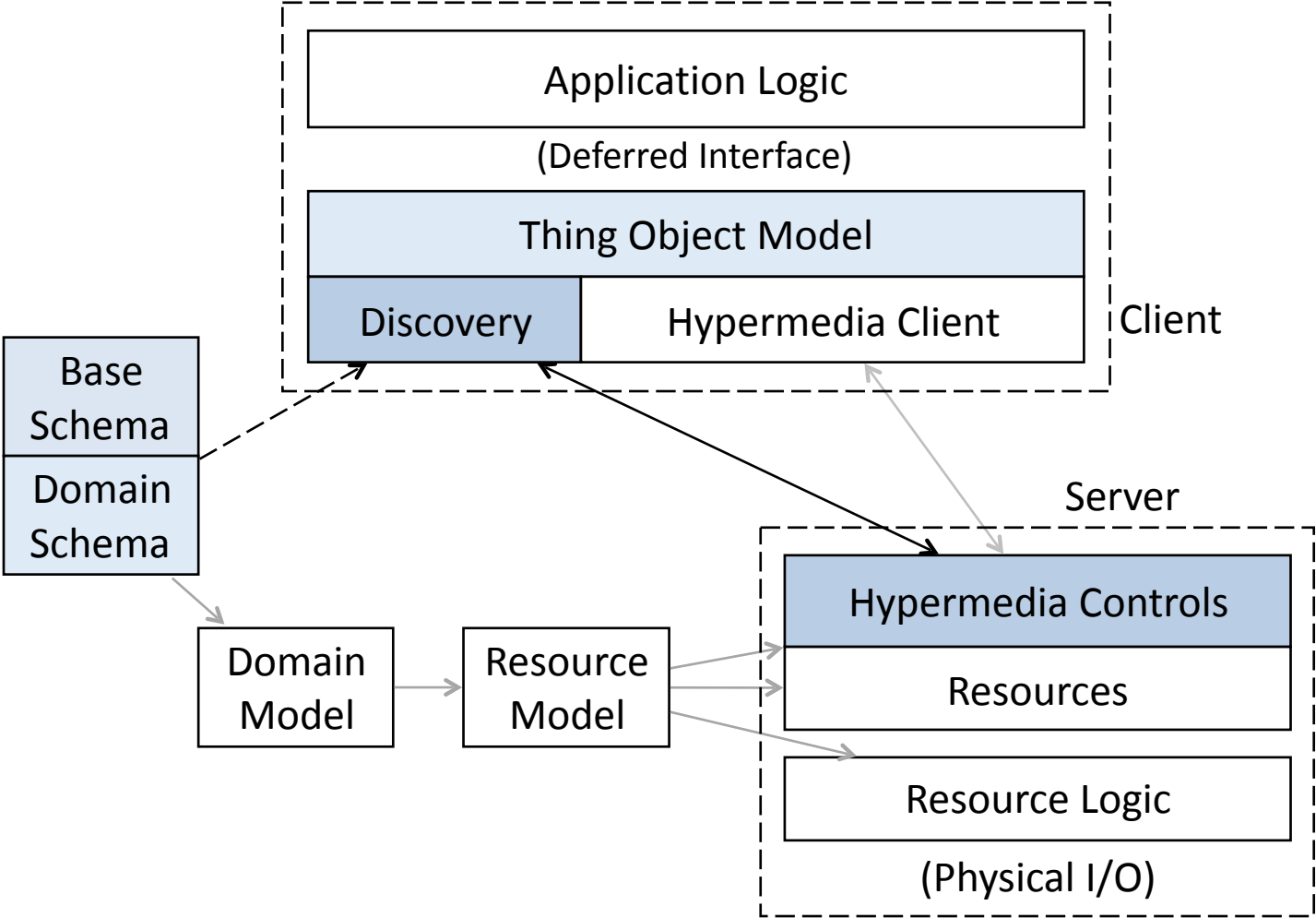
- Existing Media Types
 - application/link-format+json (**discovery**)
 - application/senml+json (**data items**)
- Extended
 - **form** values for SenML
 - application/collection+senml+json (**item composition**; embed, link)
- JSON-LD context
 - <http://thingschema.org>



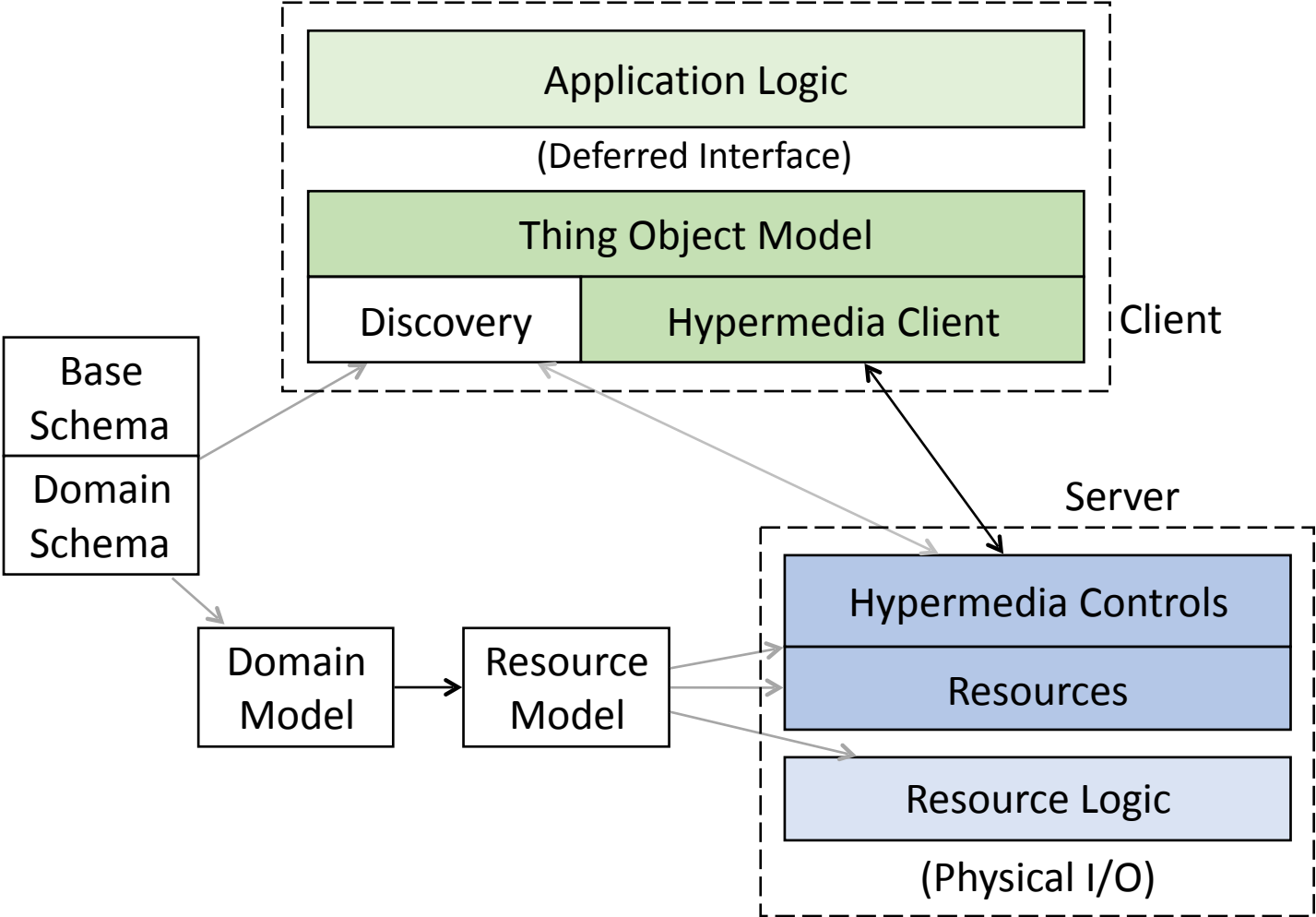
Hypermedia System Architecture



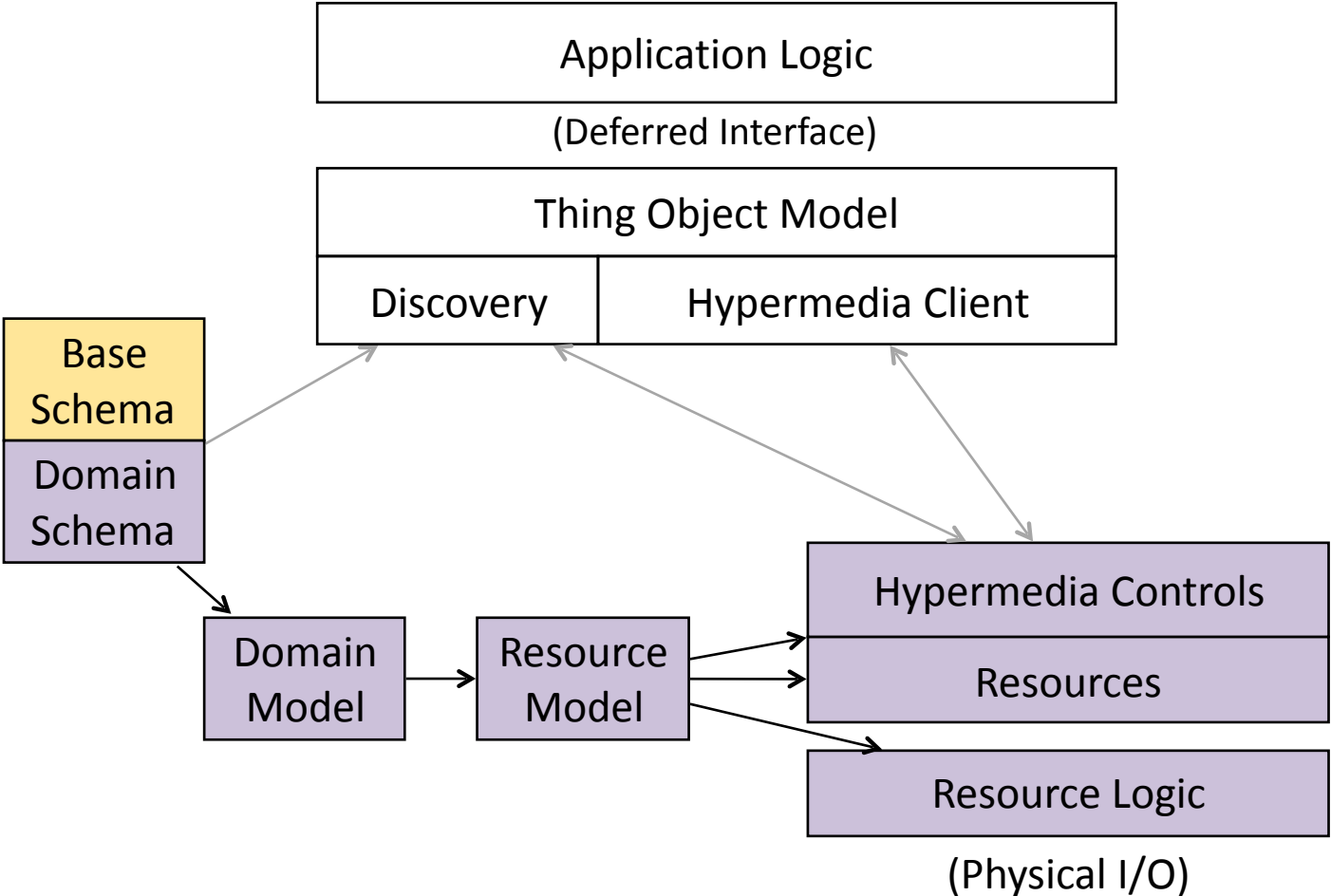
HATEOAS-driven Discovery



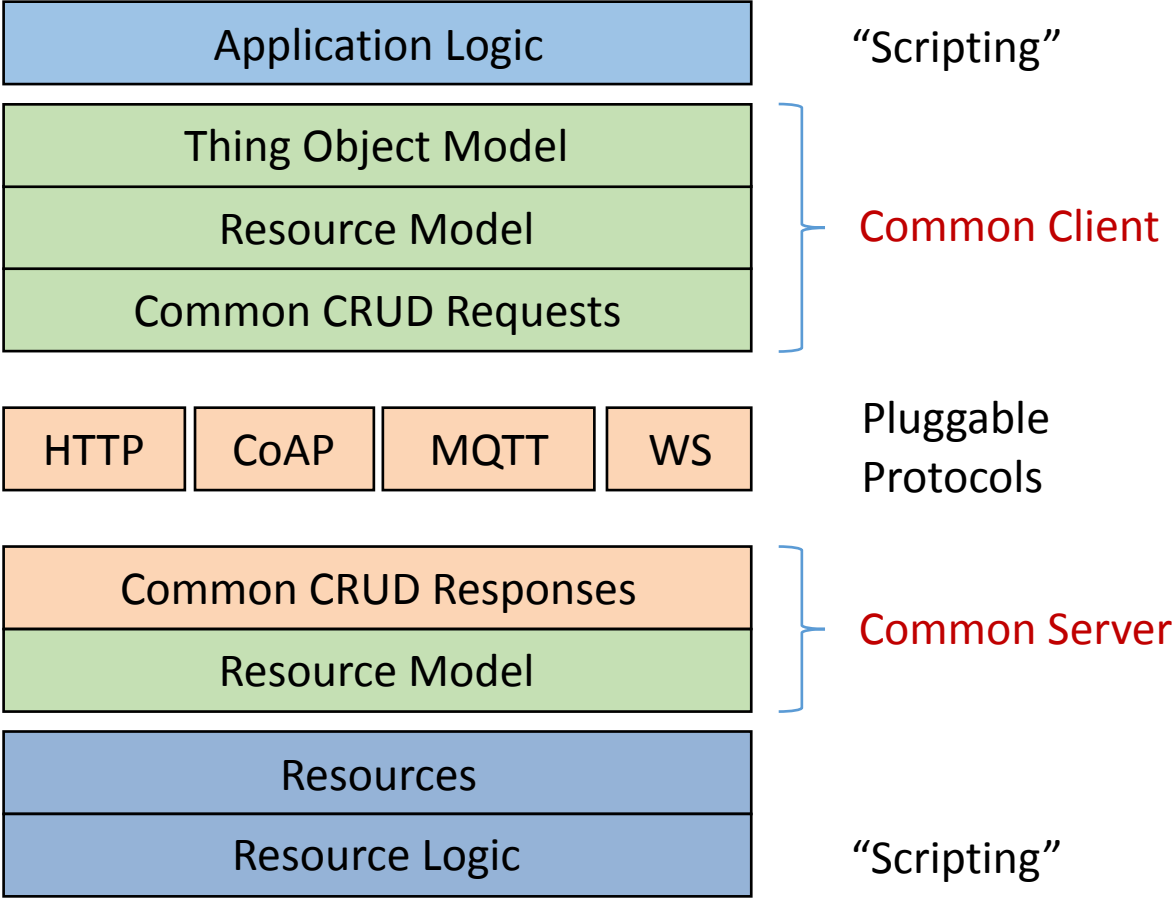
HATEOAS-driven Interaction



Model-driven Resource Construction



Modular “Runtime-ready” Architecture



Links

- <https://github.com/connectIOT/MachineHypermediaToolkit>
- <https://github.com/connectIOT/HypermediaDemo>
- thingschema.org