Research Topics in Machine Hypermedia System Design

IRTF Thing to Thing Research Group

March 15, 2016

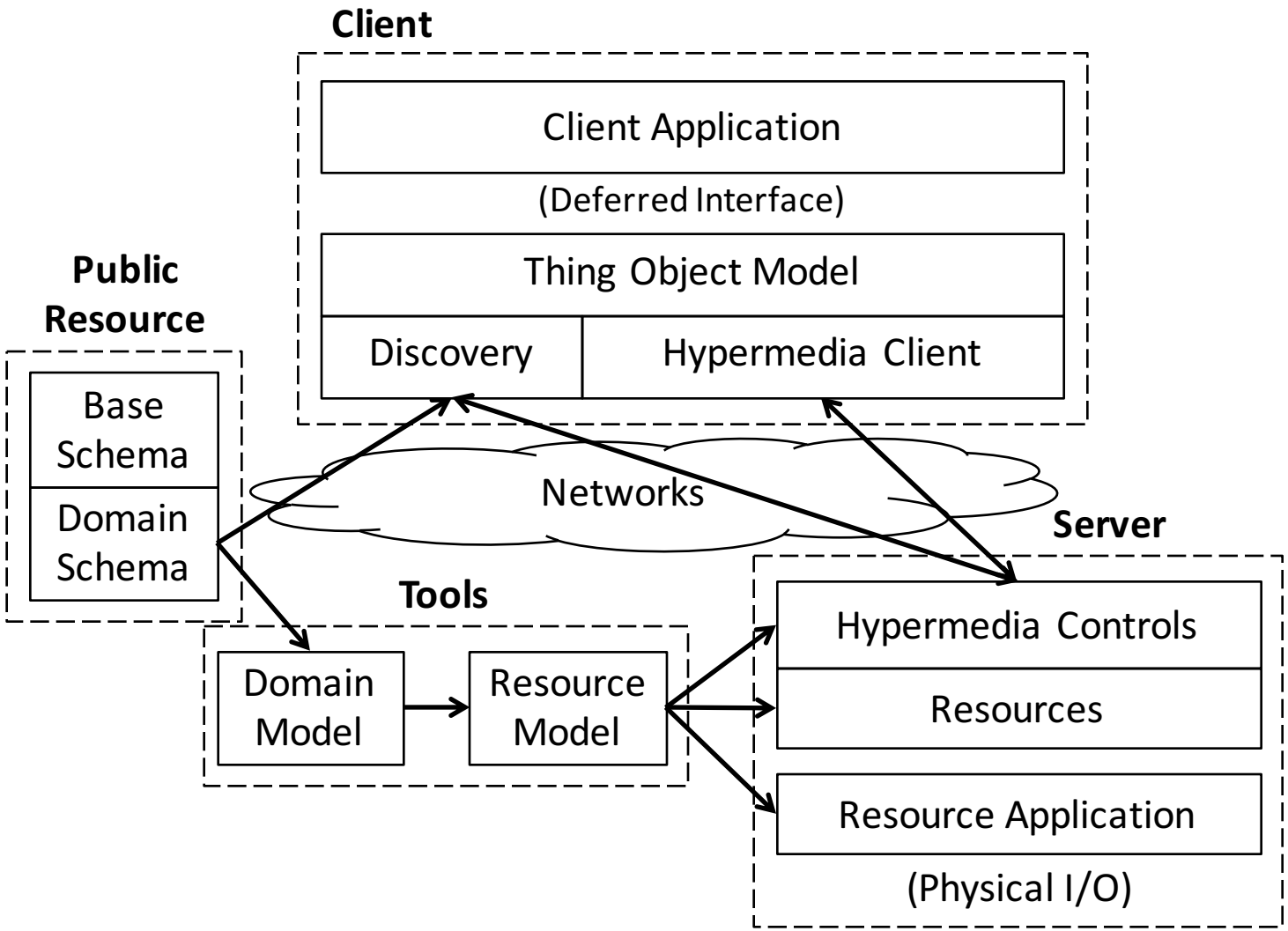
Research Goals

- Investigate design patterns for interoperable hypertext driven machine interfaces
- Investigate RESTful design patterns for sensing and actuation
- What is the role of modeling and ontology in semantic interoperability?

Research Topics

- Reference System Architecture
- Content Format Design
- RESTful Asynchronous Communication
- RESTful Actuation
- REST Protocol Abstraction
- Model Based Hypertext Annotation

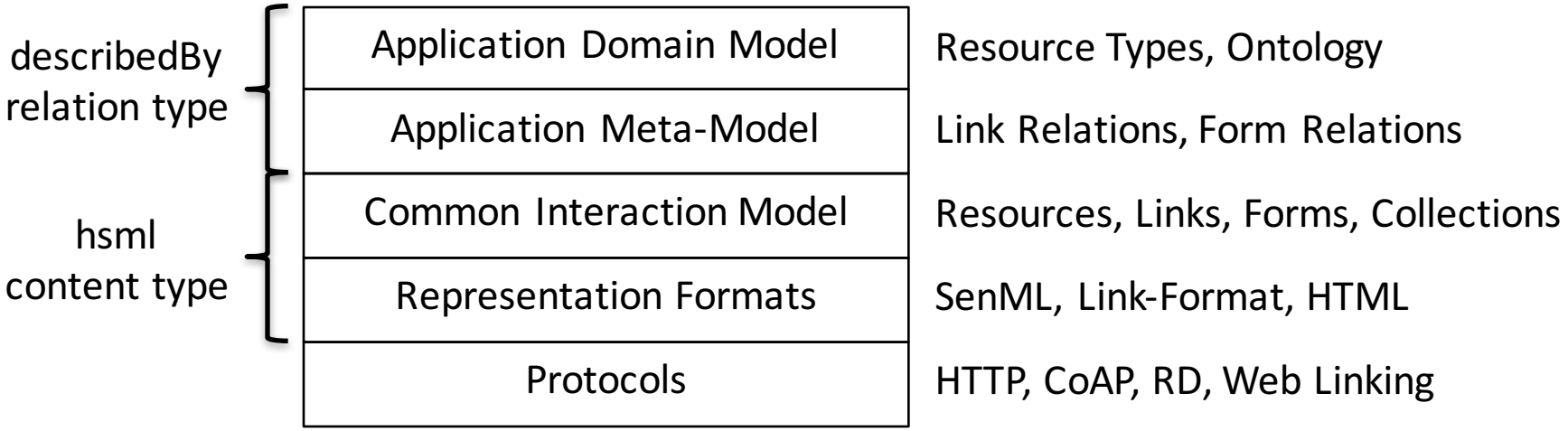
Reference Architecture



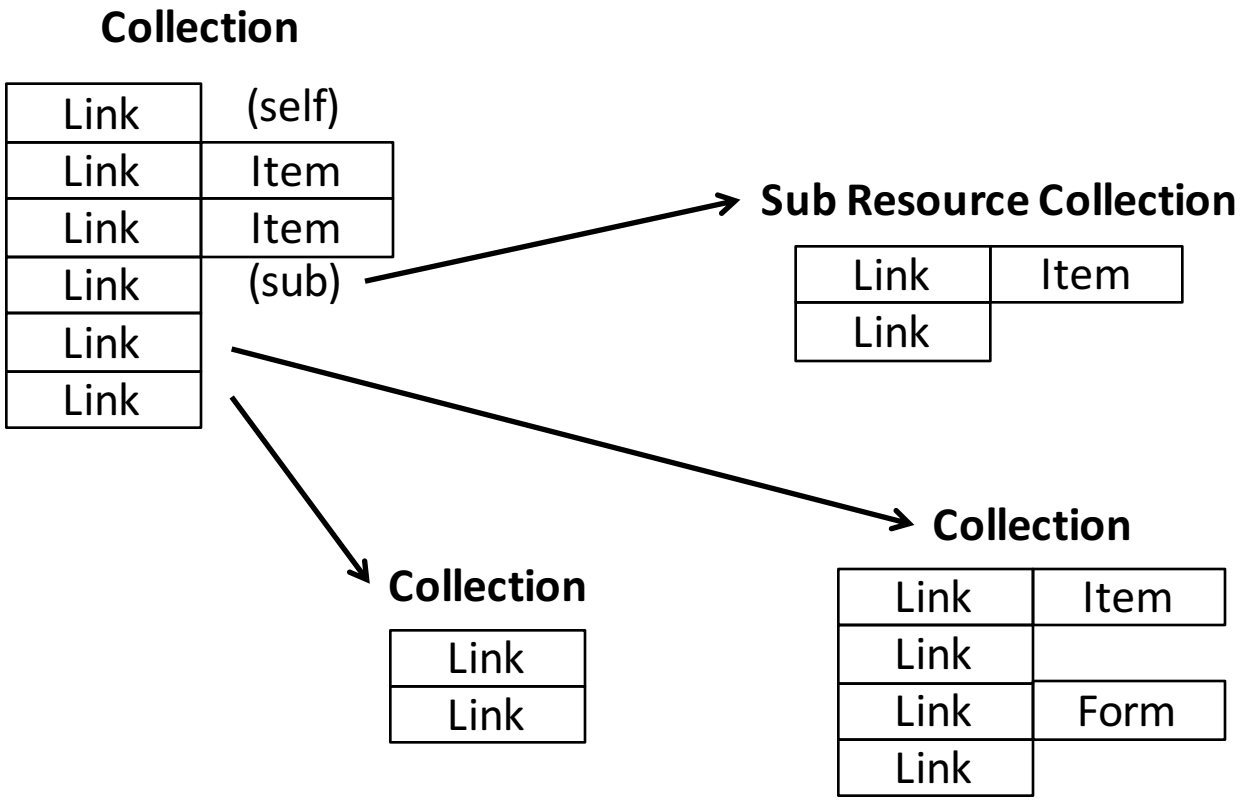
Content Format Design

- What should a content-format describe?
- Representation Formats
- Common Interaction Model
  - Resource model e.g. CoRE Interfaces
  - Data model e.g. SenML
  - How links work, e.g. CoRE Link-Format
  - How forms work
  - How relation types are used
  - Some base relation types

Content Format Design



Resource Model



Representation Format Example

```
{
    "bn": "/light/onOff/currentState/",
    "e": [
      {
        "vb": false,
        "n": ""
      }
    ],
    "l": [
      {
        "href": "",
        "rel": ["self","item"],
        "rt": ["property", "currentstate"],
        "ct": ["application/senml+json"]
      }
    ]
}
```

Items may be represented in SenML

Links may be represented in CoRE link-format

Forms

To "invokeAction" of type "change" on the "/light3/brightness/" resource, perform a "post" to the resource at "/light3/brightness/actuations" using the "application/hsml+json" content format

```
{
  "anchor": "/light3/brightness/",
  "rel": "invokeAction",
  "type": "change",
  "method": "post",
  "href": "actuations",
  "accept": "application/hsml+json",
}
```

Link and Form Relations

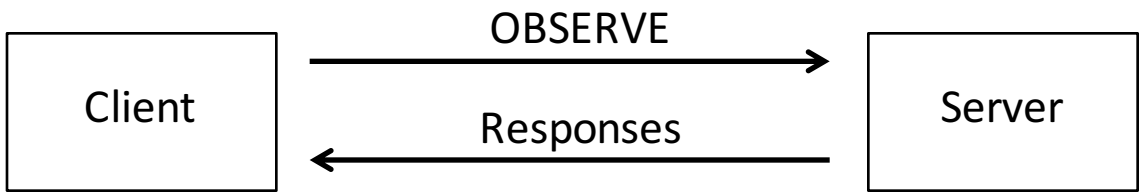
- Link Relations
  - "item" (an embedded item in a collection)
  - "sub" (a subresource item in a collection)
  - "form" (a form item in a collection)
  - "grp" (a group interaction link)
- Form Relations
  - "addItem" (add an item to a collection)

RESTful Asynchronous Communication

- REST interaction is a state machine between client and server – request and response
- Asynchronous Communication using REST is one or more state transition responses that take place after a request is made
- Two classes of interaction:
  - Between resources and applications
  - From Resource to Resource

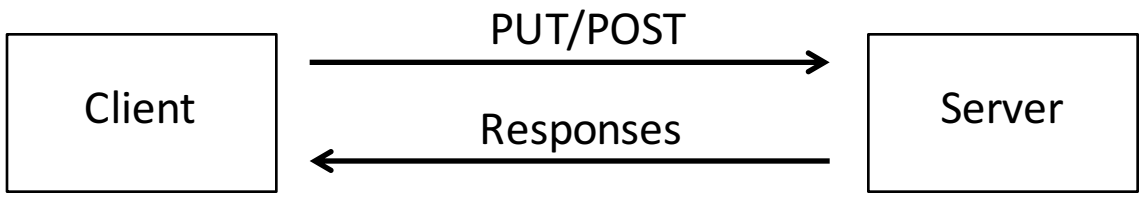
Resource to Application

- CoAP Observe is a RESTful asynchronous communication method
- Client application makes state changes based on server responses
- Server is the name and state origin of the resource



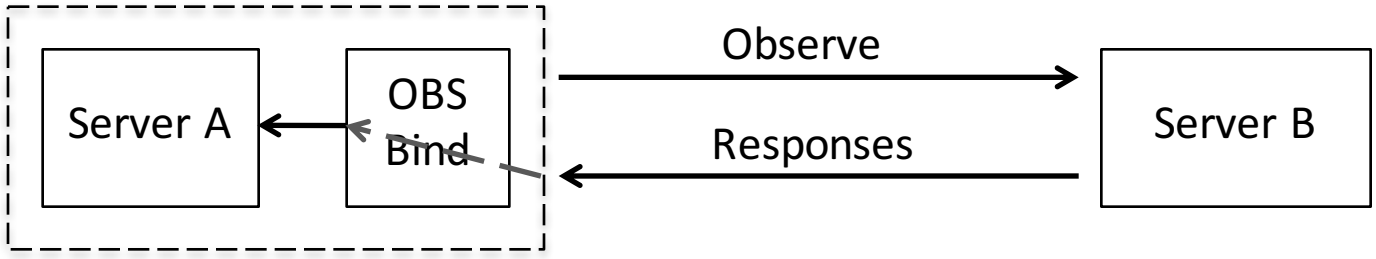
Resource to Application

- PUT or POST can be used for the client to update the state of the server
- Client application changes state on server asynchronously
- Server is the name and state origin of the resource



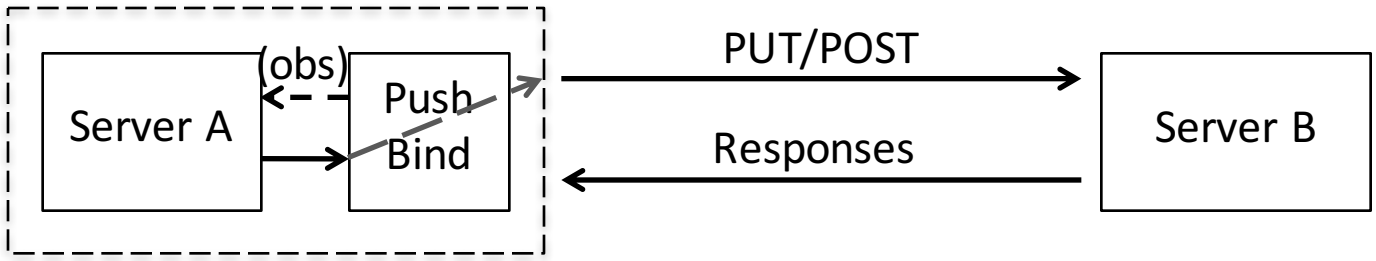
Resource to Resource

- A client instance may be "bound" to a resource and perform state transfer between it and another resource
- Observe binding updates the state of the locally bound resource based on responses from the "boundTo" resource
- Server B is the name and state origin of the resource



Resource to Resource

- Push binding observes the state of the locally bound resource and updates the "boundTo" resource
- Local binding may incorporate a filter and may be forms-capable
- Server B is the name and state origin of the resource



Notification Resource

- Collection resource to capture notifications from a binding
- Binding uses POST to create a new resource in the collection for each notification
- Client Application can observe the collection for new notifications being created and receive representations
- Server A is the name and state origin

Promise+

- Pattern for application scripts to handle recurring events like notifications from resource observations
- Extends the Promise pattern with an update handler for recurring events

```
res.observe().then(onResolve, onReject, onUpdate)
(…do other stuff)

onUpdate(value){
  processStateUpdate(value)
```

HTTP Observe

- Using a technique based on HTML5 Server Sent Events (SSE)
- Header "Transfer-Encoding:chunked" enables open TCP connection to be used for asynchronous messages
- Messages could be formatted as HTTP Response and header lines, with content-type and content-length controls
- Header options e.g. Observe:0 could be used to create a CoAP-compatible observe for HTTP

RESTful Actuation

- What is RESTful actuation?
- Change of state on a resource that has some effect in the physical world
- Many different interpretations of this:
  - Update of a resource directly changes physical state
  - Update of a resource communicates intended state
  - Creation of a resource that describes the intended state transition
  - Update of a setpoint resource of a controller

Update State Resource

- There is always some uncertainty, e.g. the physical process may fail or be delayed
- Will the state returned on a subsequent read reflect the intended state or the actual state?
- Intended state is technically RESTful but not useful
- Actual state is useful but not RESTful
- Delaying the response until intended state is observed might work…

Update Intended State Resource

- This will be both RESTful and Useful
- Allows a REST response to be generated for the intended state and application can then monitor observed state
- This could work, but what about where we want to parameterize execution with transition times, etc? How do we know if the action is going to succeed or fail?

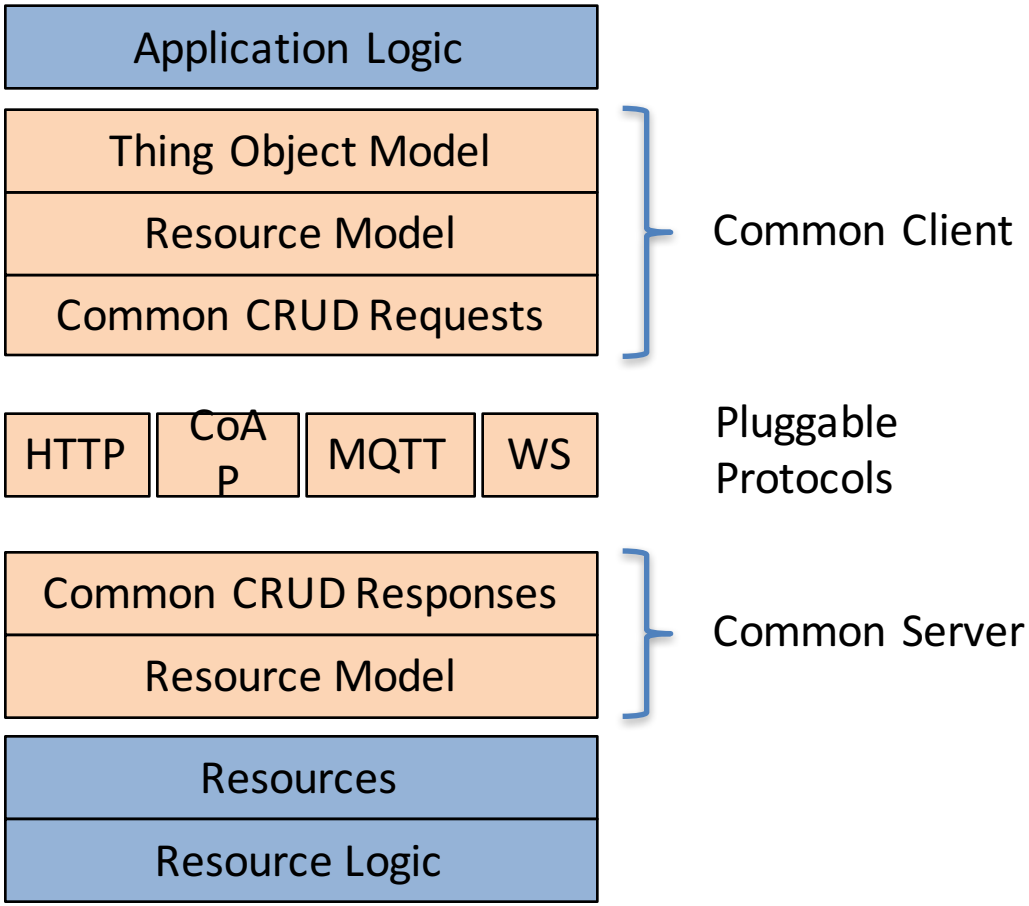
Create a State Transition Resource

- This is RESTful and useful, using the resource create pattern with a form, and returning a resource location that can be monitored for state changes
- Allows asynchronous notification and promise to be used to track progress, success, failure of running actions, also to modify or cancel
- Multiple actions may be queued

Controllers

- Thermostat is an example of a controller
- Temperature setting is a set-point that is input to a controller algorithm that decides whether to operate an actuator based on the relationship of the set-point to the measured temperature, and perhaps other variables
- Thermostat has measurement temperature and set-point temperature inputs, and an actuator state output.

REST Protocol Abstraction

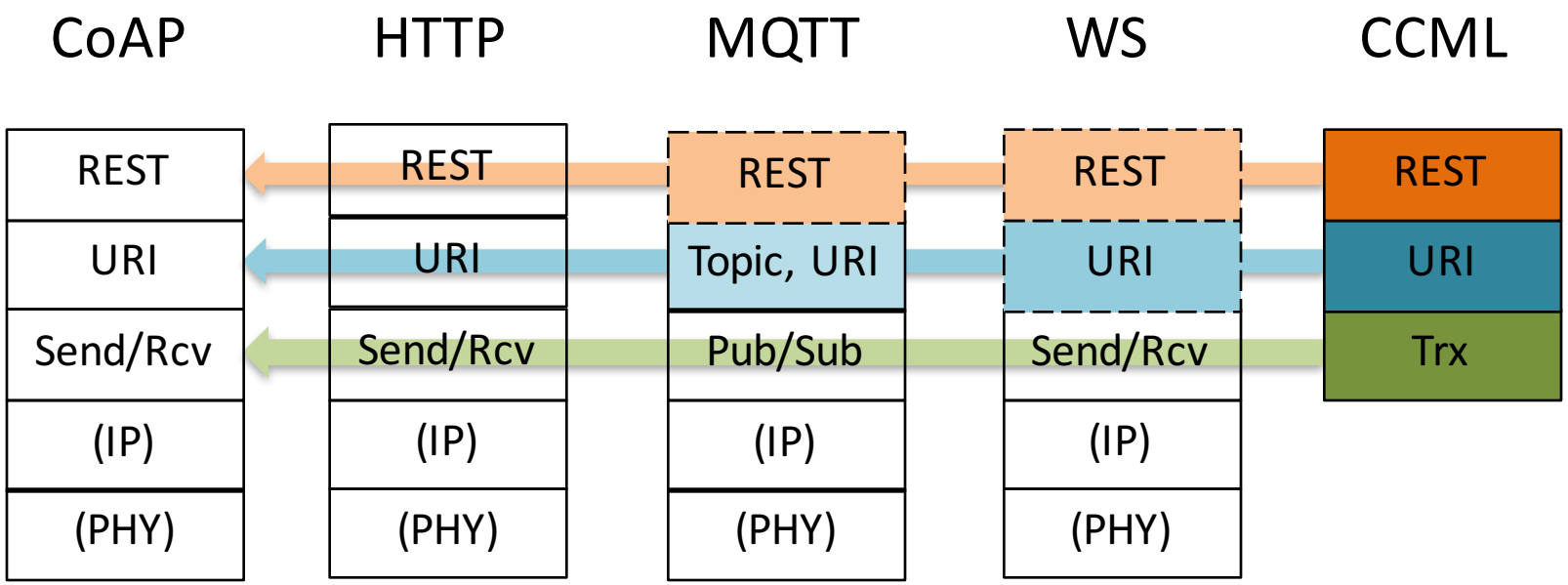


Dictionary Mapping of Common REST Transaction Layer

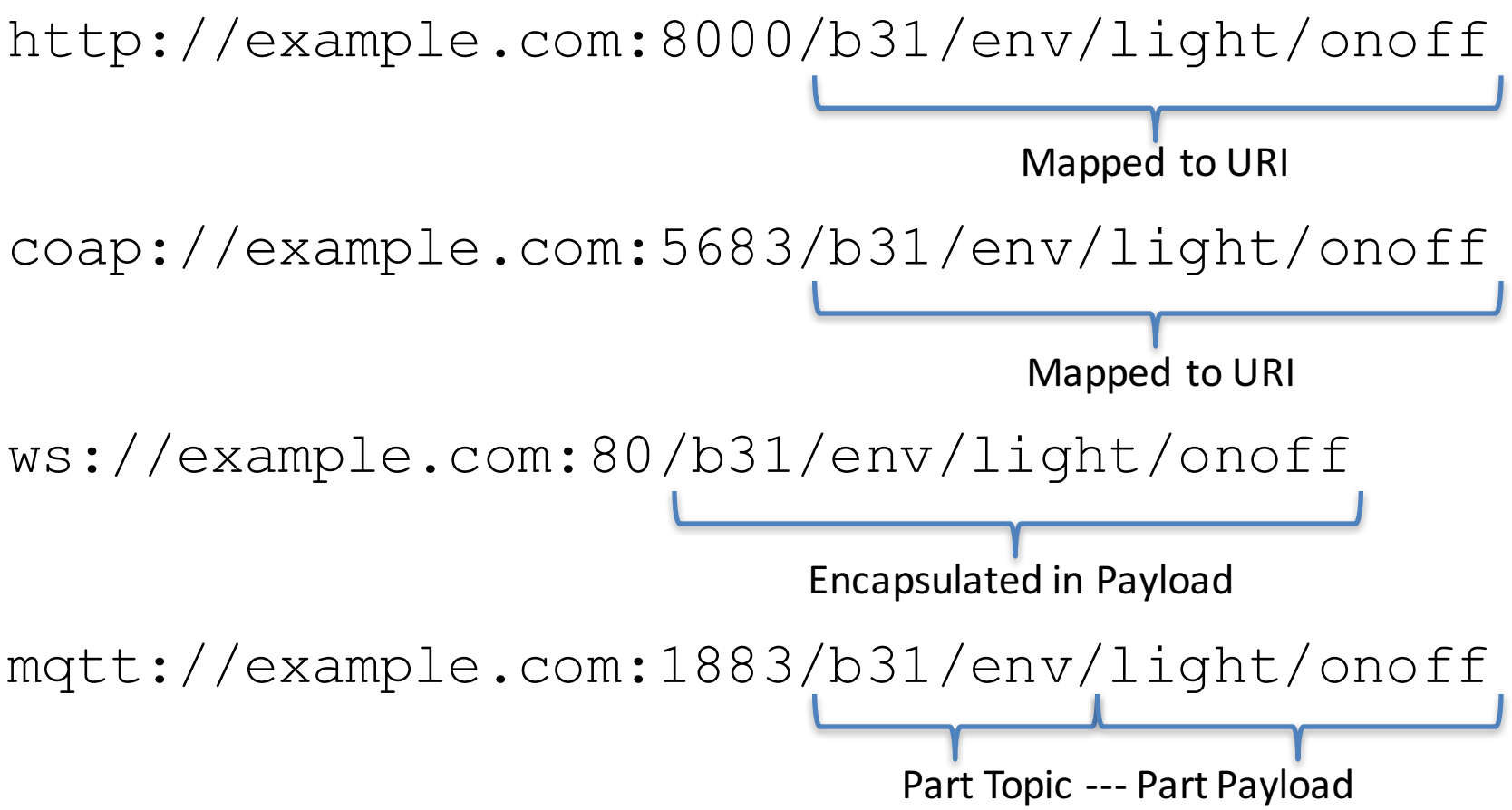

```
{
"uriPath": ["/","a", "b"],
"uriQuery": {"rt": "test", "obs": "true"}
"contentFormat": "application/link-format+json",
"options": {}
"method": "GET",
"payload": null,
"response": {
    "status": "Success",
    "contentFormat": "application/link-format+json",
    "payload": "[{"href":"","rel":"self","rt":"test"}]"
    }
}
```

Common CRUD

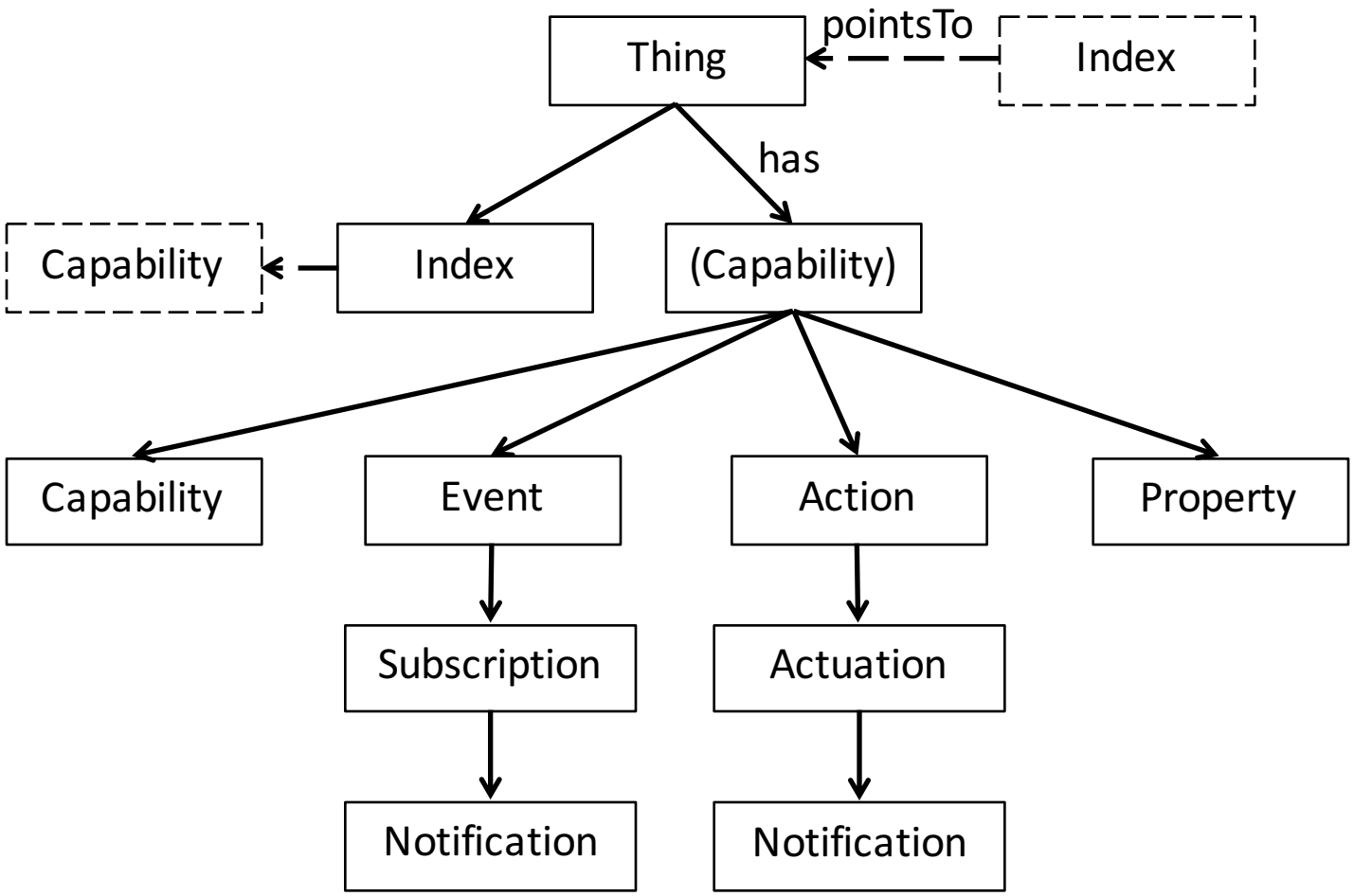
- Map abstraction to HTTP and CoAP request and responses
- Encapsulate the abstraction in WS and MQTT payloads



Consistent Resource Identifiers: Cross-Protocol Hyperlinking



Model Based Hypertext Annotation



Domain Schema and Model

- Reusable terms with mayHave and usedBy definitions
  - “brightness” is used by “light” but not “motion sensor”
  - “brightness” may have “change” action but not “open”

```
class: brightness,
type: capability,
description: “brightness control”
usedBy: [ light ],
mayHave: [
    currentBrightness, targetBrightness,
    stepBrightness, moveBrightness,
    change, step, move, stop,
    propertyValueChange ],
params: {_
    targetValue: _targetBrightness,
   _stepSize: _stepBrightness,
   _moveRate: _moveBrightness},}
```

Domain Model Example

```
"@context": "http://thingschema.org",
"resource": [
  {
    "type": "light",
    "name": "light",
    "capabilities": [
        {
          "type": "brightness",
          "name": "brightness"
        },
        {
          "type": "onoff",
          "name": "onoff"
        }
```

Demonstrator and Reference Implementation

- Machine Hypermedia Toolkit is an open source reference implementation

https://github.com/connectIOT/MachineHypermediaToolkit

- Demonstrator resource on Github for tutorial introduction

https://github.com/connectIOT/HypermediaDemo

Resources…

- These slides

http://www.slideshare.net/MichaelKoster/research-topics-in-machine-hypermedia

- Blog Article

http://iot-datamodels.blogspot.com/2015/10/hypermedia-design-for-machine-interfaces.html

- Demo Resource

https://github.com/connectIOT/HypermediaDemo

- Reference Implementation (work in progress)

https://github.com/connectIOT/MachineHypermediaToolkit

- CoRE Interfaces

https://datatracker.ietf.org/doc/draft-ietf-core-interfaces/

- Link-Format

https://tools.ietf.org/html/rfc6690 , https://tools.ietf.org/html/draft-ietf-core-links-json-04

- SenML-01

https://datatracker.ietf.org/doc/draft-jennings-core-senml/01/