Discovery for the Web of Things

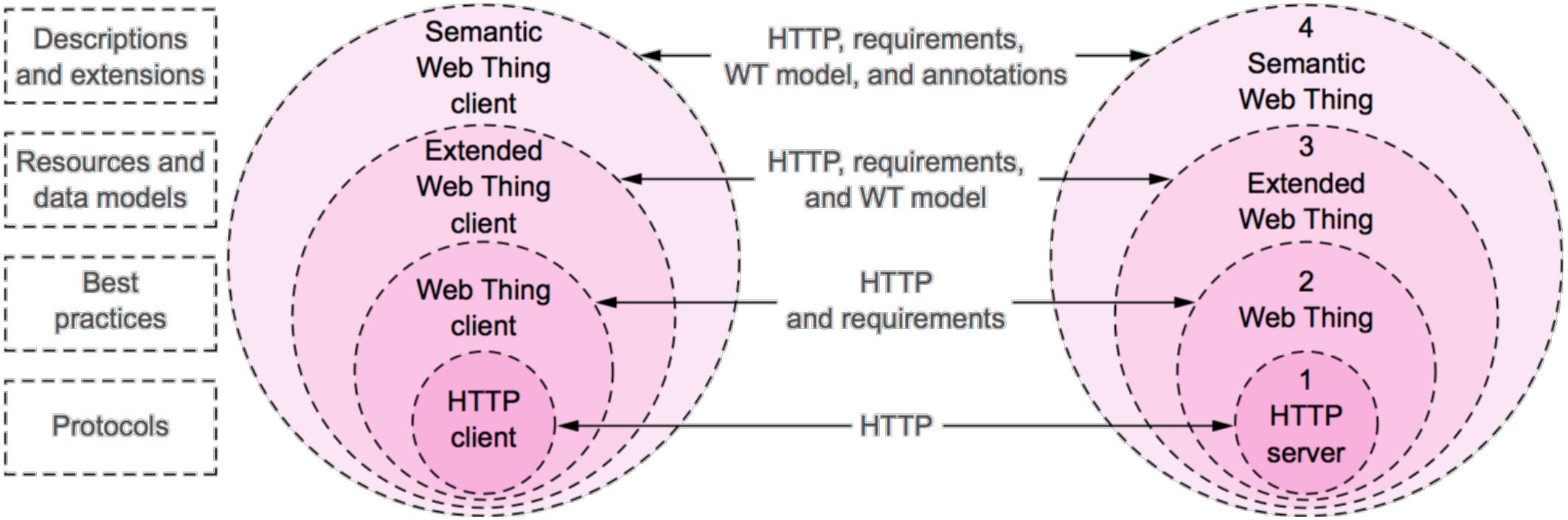
Niels Olof Bouvin

The Challenge of Interoperability

JOKE!

- **What if… there was a *fifth* milestone?**
  - "integrate all other groups' Things into your P2P storage system" . . . due Friday!
- **How would you go about such a task?**
  - collect the URL of each Thing, and
  - inspect each Thing's API and write custom code for everyone?
  - would *you* want to have to rely on the API documentation that *you* have written?
- **There *has* to be a better way**
  - but we would need to come to a shared understanding and practice for it to work

Levels of specification



- **We can vary how deeply we specify our Things**
  - it's more work, but it might ease interoperability
- **Ideally, it would help make our Things more robust**
  - possible to, e.g., verify their API, which is excellent for testing purposes

What do we need?

- **A way to discover local Things**
- **A way to discover the Things' API programmatically**
  - sufficient well to know required arguments, responses, etc.
- **A way to discover Things across the Internet through search engines**

Overview

- **Discovering Things**
- Towards a general model for Things
- The Semantic Web of Things

Is there any Thing out there?

- **HTTP does not really have any discovery mechanism**
  - you are expected to know your destination's address from *somewhere*, often a link
  - once you have reached a page, you can usually navigate to the rest of the site, but this is strictly based on conventions
  - over time, we have grown accustomed to search engines indexing sites, bypassing the need for a proper discovery mechanism
- **So, how might we discover which Things are present?**

Discovery services for IP

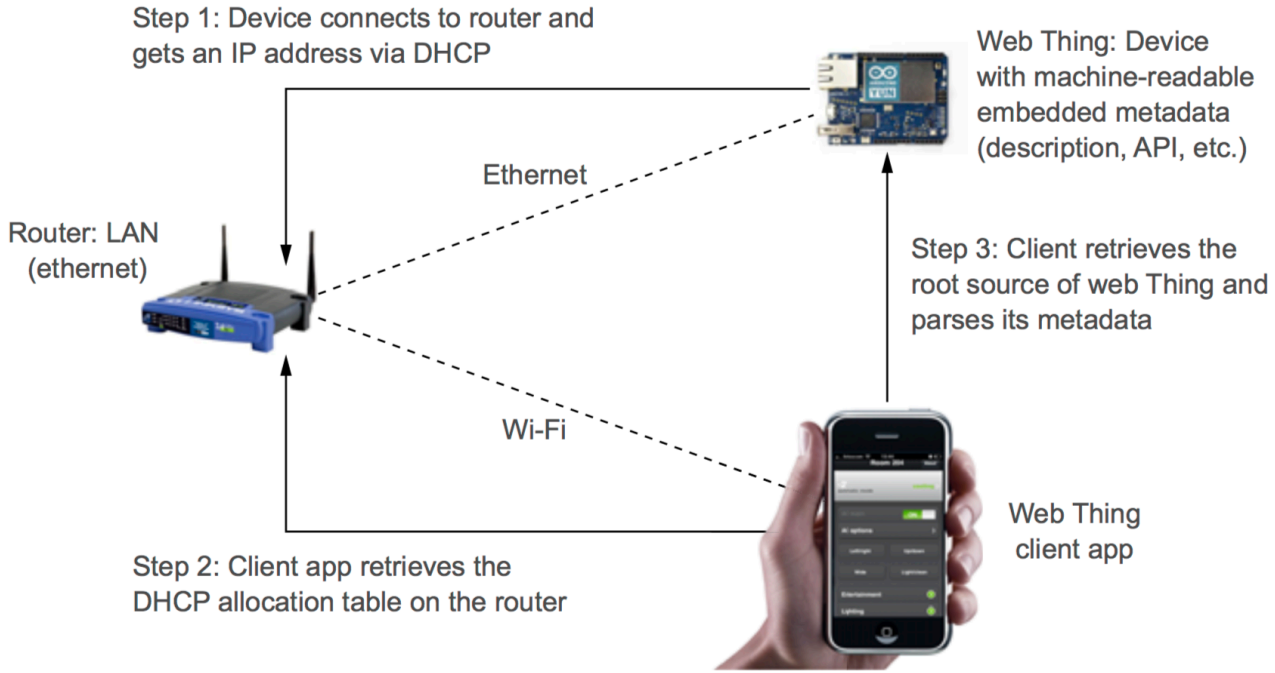
- **Discovering new devices on the local network is not a new problem. Often referred to as zero-configuration**
  - DHCP
  - UPnP, DLNA
  - mDNS, Bonjour (macOS), Avahi (Linux)
- **Usually handled through a broadcast across the local network, where devices can announce or identify themselves**
  - to announce/provide services or to search/request for services

Problem…

- **These standards are not well supported in Web browsers**
  - *used* to just work in macOS Safari, but no more!
- **So what do?**
  - install an extension?
  - have the sensor register itself somewhere and enquire there?

Possible solutions

- **Have the Things announce themselves**
  - using whatever zeroconf method and discover these announcements with an app
  - stick an QRCode on the device; have it announce its presence as a Bluetooth LE beacon
- **Augment the router to share DHCP addresses in JSON**
  - this is a hack: determining the router's address is going to be guesswork



Overview

- Discovering Things
- **Towards a general model for Things**
- **The Semantic Web of Things**

Making your Thing indexable

- **The vast majority of all Web sites are discovered through search engines**
  - they crawl sites by following links, index their contents, and make them discoverable through search interfaces
  - not a terribly elegant solution, but it works pretty well
- **If we want our sensor to be discoverable on the Web, it *must* be indexable by search engines**
  - we could provide HTML pages for all endpoints, making it possible to crawl and index these pages
  - but what about tiny Things? What about JSON and other unHTML formats?
- **Plus, a well-understood structure will ease coding**

IETF RFC5988

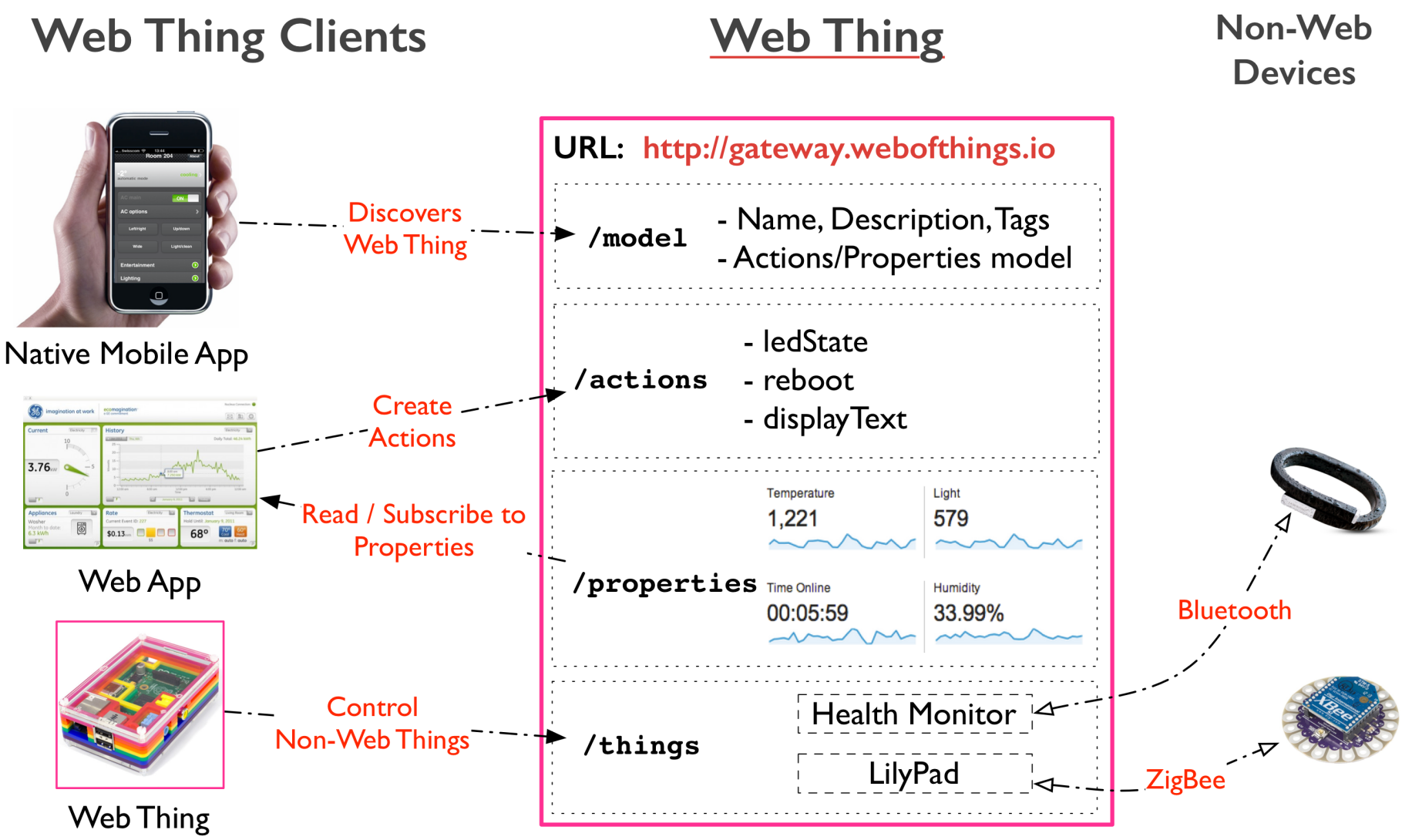
```
–> Request
HTTP 1.1 GET /
Host: MyLittleThingie.io
Accept: application/json

<– Response
200 OK
Link: </model/>; rel="model", </properties/>; rel="properties", </actions/>;
      rel="actions", </things/>; rel="things", <http://model.webofthings.io/>;
      rel="type", </help>; rel="help", </>; rel="ui"
```

- **A format to communicate relationships between Web resources directly in the HTTP header**
- `</model/>` **points to the URL** `mylittlethingie.io/model`**, and** `rel="model"` **indicates the role of the page**
- **This is already supported in in the <head> tag in HTML, but RFC5988 makes it possible to extend this to all general resources accessible over HTTP**

The Web of Things model

- **If we are to communicate freely with Things, we need a shared model between them**
- **This model must encompass all possible Things**
  - from a simple product ID in a tag, e.g., on a milk carton
  - over sensors
  - to systems with actuators with complex rules

The Web Thing Model



- **A general model for what a Web Thing actually is**

The Web Thing Model: /

- **The root resource of the Thing, which is the virtual representation of the Thing itself**
  - `{wt}` is the complete URL
- **It can, of course, be altered in the usual fashion**

```
-> REQUEST
GET {wt}
Content-Type: application/json

<- RESPONSE
200 OK
Link: <model/>; rel="model"
Link: <properties/>; rel="properties"
Link: <actions/>; rel="actions"
Link: <product/>; rel="product"
Link: <type/>; rel="type"
Link: <help/>; rel="help"
Link: <ui/>; rel="ui"
Link: <_myCustomLinkRelType/>; rel="_myCustomLinkRelType"

{
  "id": "myCar",
  "name": "My super great car",
  "description": "This is such a great car.",
  "createdAd": "2012-08-24T17:29:11.683Z",
  "updatedAd": "2012-08-24T17:29:11.683Z",
  "tags":["cart", "device", "test"],
  "customFields":{
      "size": "20",
      "color":"blue"
  }
}
```

The Web Thing Model: /properties

- A Property keeps track of a set of variables about a device (its location, the temperature sensor reading, etc.)
- The individual properties can also be accessed individually
  - as well as their history, so our architecture differs here

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Link: <http://model.webofthings.io/#properties-resource>;

{
    {
       "id": "temperature",
       "name": "Temperature Sensor",
       "values": {
           "t": 9,
           "timestamp": "2016-01-31T18:25:04.679Z"
       }
       "id": "humidity",
       "name": "Humidity Sensor",
       "values": {
           "h": 70,
           "timestamp": "2016-01-31T18:25:04.679Z"
       }
       "id": "pir",
       "name": "Passive Infrared",
       "values": {
           "presence": false,
           "timestamp": "2016-01-31T18:25:04.678Z"
       }
       "id": "leds",
       "name": "LEDs",
       "values": {
           "1": false,
           "2": false,
           "timestamp": "2016-01-31T18:25:04.679Z"
       }
    }
}
```

The Web Thing Model: `/actions`

- **The interface to change the state of various properties of the Thing**
- **Decouples changing properties directly**
  - like accessing an object's state through getter/setters rather directly modifying a field

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Link: <http://model.webofthings.io/#actions-resource>;
      rel="type"

[{“id”:”ledState",
  "name":"Changes the status of the LEDs”}]
```

```
–> REQUEST
POST {WT}/actions/ledState
Content-Type: application/json
{"ledId":"3","state":true}

<– RESPONSE
HTTP/1.1 204 NO CONTENT
```

The Web Thing Model: /model

- Collects all information about the Thing
- Based on this, we should be able to know what a Thing, which resources it has, and how they may be accessed and modified, including the type of arguments/results

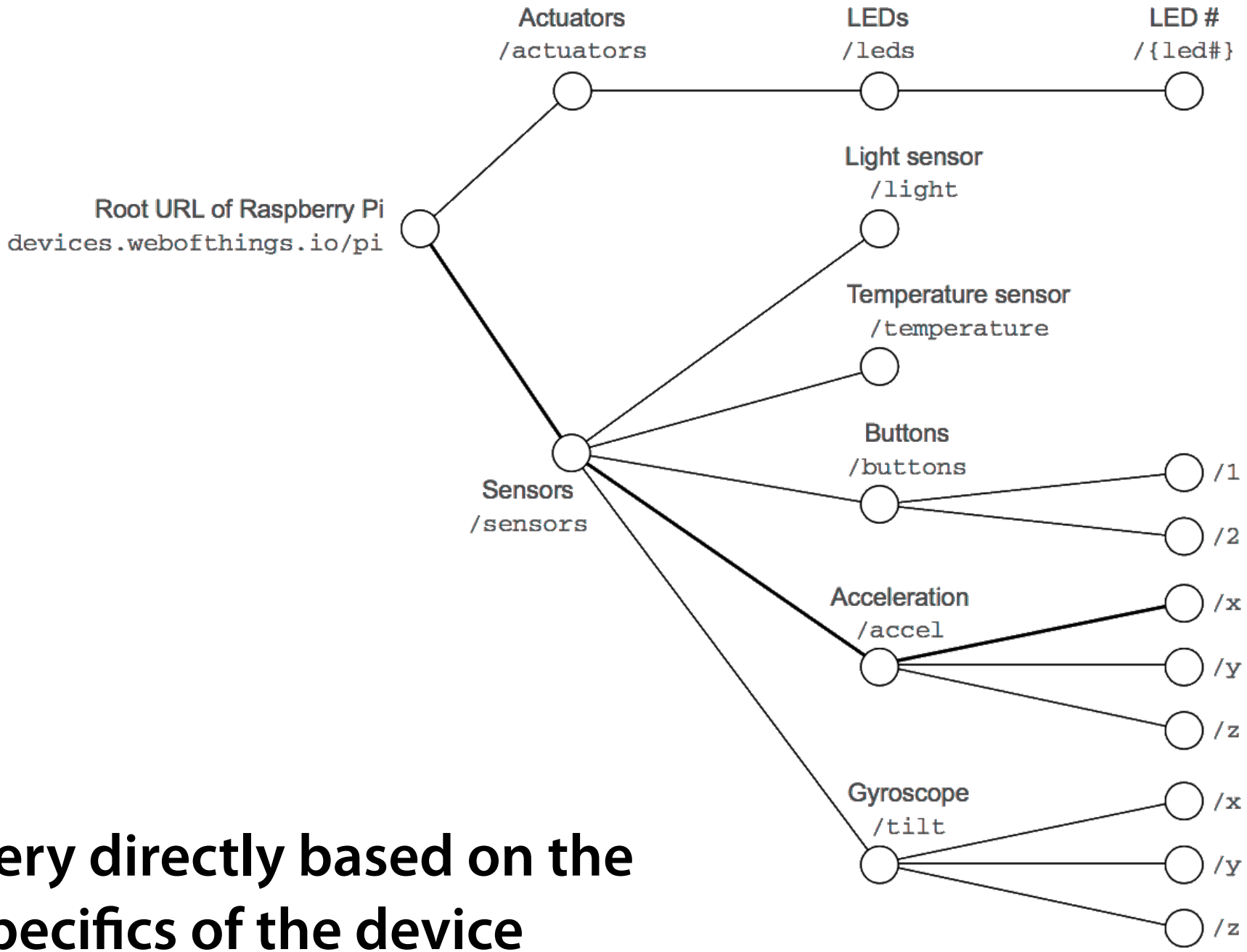
```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Link: <model.webofthings.io>; rel="type"
...

"actions": {
    "link": "/actions",
    "title": "Actions of this Web Thing",
    "resources": {
       "ledState": {
           "name":"Changes the status of the LEDs",
           "values": {
              "ledId": {
                  "type": "string",
                  "required": true},
              "state": {
                  "type": "boolean",
                  "required" : true}
           }
       }
   }
},
```

The Web Thing Model: `/things`

- **The Things that this particular Thing is a gateway for**
  - in this case a webcam and a Hue lamp
- **These 'sub'-Things can be contacted through the Thing**

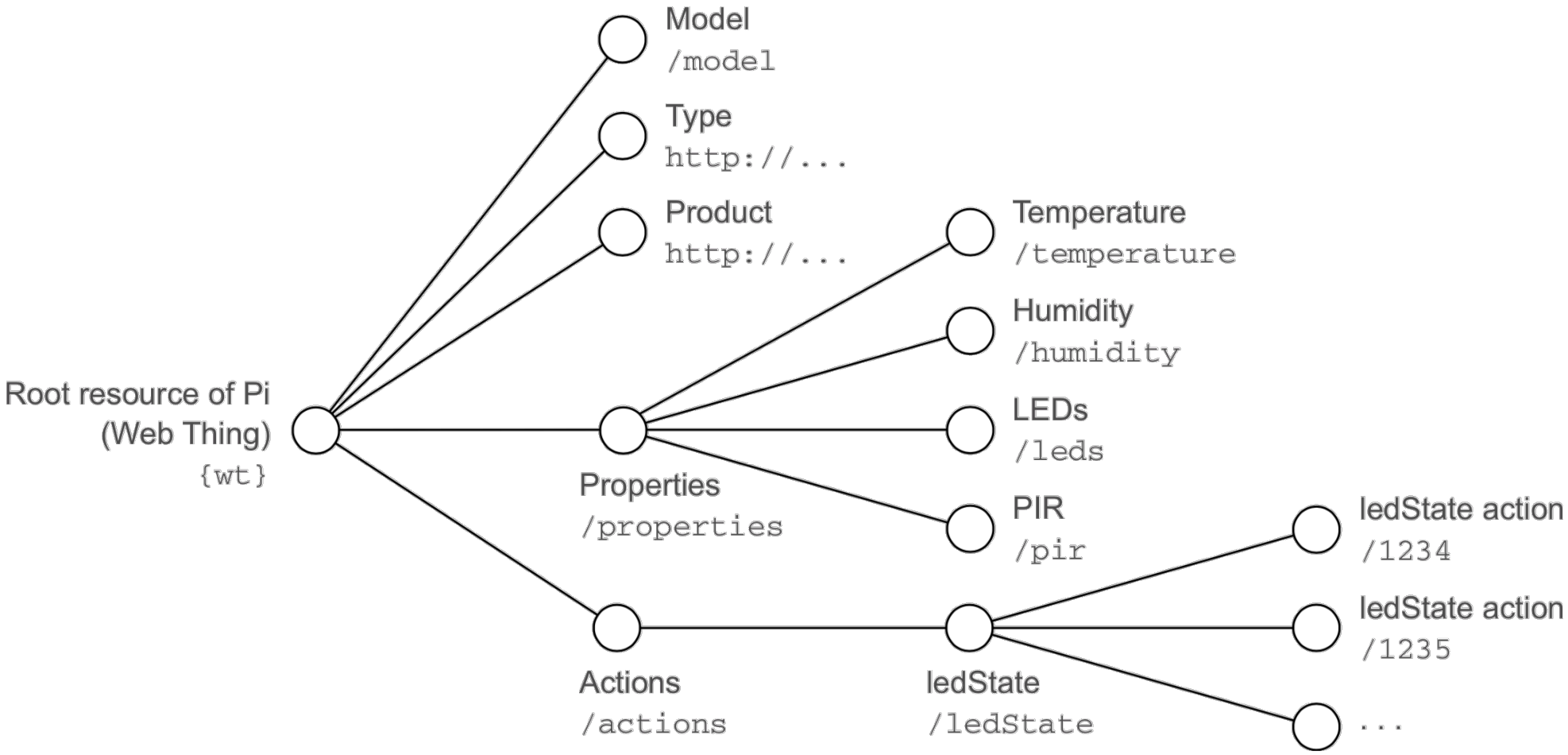
```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Link: <model.webofthings.io/things>; rel="meta"

{
    {
   "id":"http://devices.webofthings.io/pi",
   "name":"Raspberry Pi",
   "description":"A WoT-enabled Raspberry Pi"
    },
    {
   "id":"http://devices.webofthings.io/camera",
   "name":"Fooscam Camera",
   "description":"LAN-connected camera."
    },
    {
   "id":"http://devices.webofthings.io/hue",
   "name":"Philips Hue",
   "description":"A WoT-enabled Philips Hue Lamp."
    }
}
```

The old model implementation



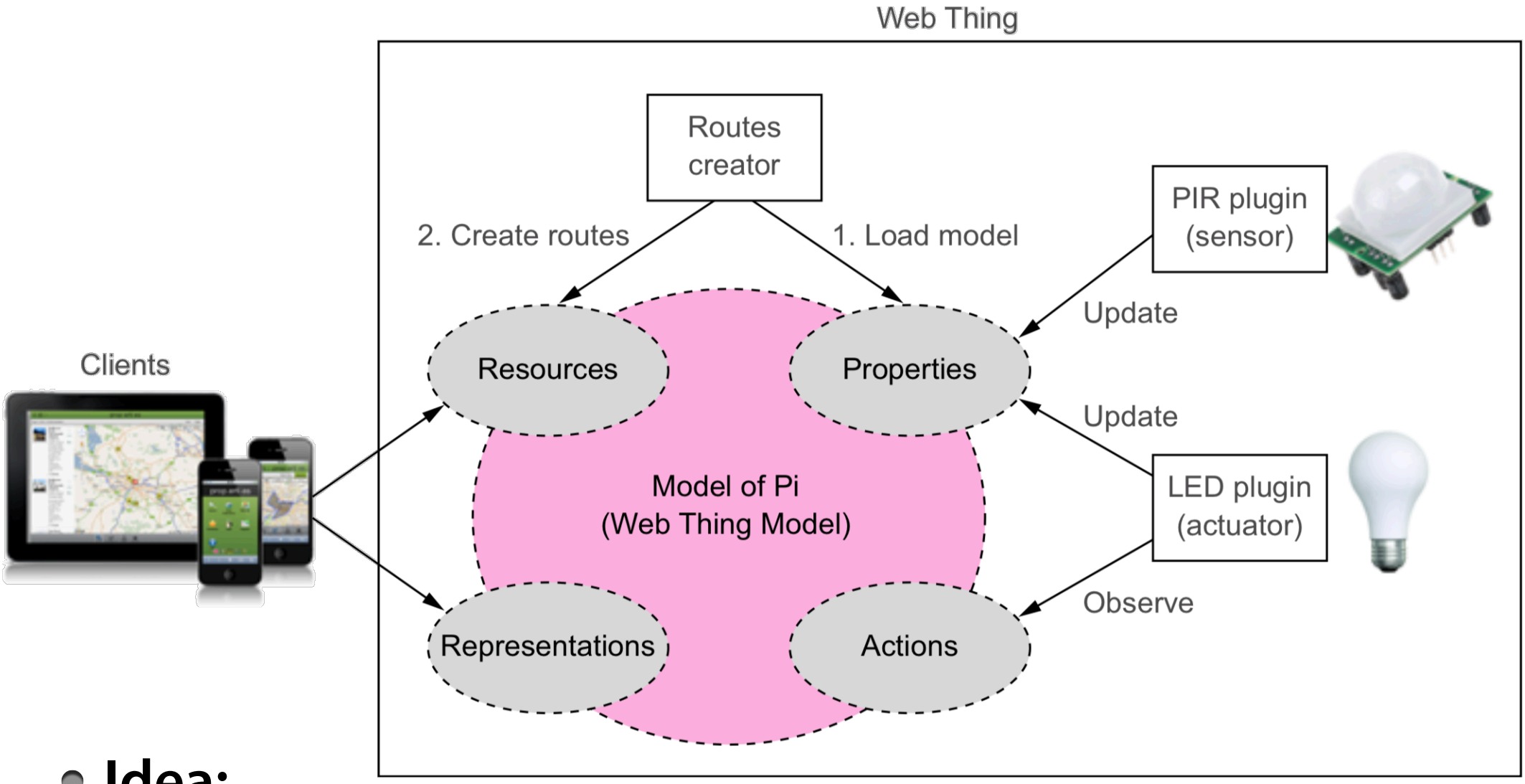
- Very directly based on the specifics of the device

Implementing the WoT model



- **A generalised model that can encompass more**
  - and, given the right framework, automatically be put to work

The model shall be our guide



- **Idea:**
  - create a comprehensive model of the Thing
  - auto generate routes, etc., based on the model

Auto-connecting code

- It can, of course, not all just be automagical
- But, with a sufficient specific model, and prepared code for sensors, it is fairly straightforward
  - see the code in `wot-book/chapter8-semantics` (available from GitHub)
- The model permits us, by providing a systematic description of the properties and associated actions of the Thing, to create a generic user interface
- Given the identities of the Things, you could, e.g., make a Web page that reported on all temperature sensors in a building

Overview

- Discovering Things
- Towards a general model for Things
- **The Semantic Web of Things**

The Semantic Web

- **An *old* project spearheaded by Sir Tim Berners-Lee, (one of) the inventor(s) of WWW, and W3C's director**
- **The Web consists of formatted text, as well as other resources**
  - while there have been great strides in making computers understand text and images, they are still not very good at it compared to humans
- **The Semantic Web would have Web resources annotated with meaning, i.e., semantics**
  - "this is a person"; "this is an address"; "this is a temperature in Celcius"; etc.
- **This would make it far easier for machines to index, and subsequently, make available through search**

Challenges for the Semantic Web

- **Who authors the metadata?**
  - what is their immediate benefit?
  - can we trust their metadata better than their data?
- **This has been a recurring problem with the vision of the Semantic Web: in order for it to work as intended, much of the Web must be annotated sufficiently well for machines to extract and understand**
  - so far, this has not been going well
- **Within specific fields of practice and use cases, it can work**
  - but it has met with limited success on a greater scale

The reality of the Semantic Web

- **If Google et al. does not understand the semantics, there is little point to bother with it**
  - The Web Things Model may be a W3C working group's work in progress, but it is not recognised by the search engines
  - *any* business model relying on Google and other search engines "soon" indexing specific semantics is overly optimistic
- **So, what *do* they index?**
  - ordinary Web pages, as you would expect, using heuristics that are top secret to extract meaning
  - some additional information, added to the Web pages, to make certain elements easier to recognise by the indexer
  - such as identifying things as Products

Providing meaning with annotations

- **HTML, as well as JSON can be annotated using various standards and associated ontologies**

```
<div vocab="http://model.webofthings.io/" typeof="WebThing">
  <h1 property="name">Raspberry Pi</h1>
  <div property="description">
    <p>A simple WoT-connected Raspberry PI for the WoT book.</p>
  </div>
  <p>ID:<span property="id">1</span></p>
  <p>Root URL:<a property="url" href="http://devices.webofthings.io">http://
      devices.webofthings.io</a></p>
  Resources:
  <div property="links" typeof="Product">
    <a property="url" href="https://www.raspberrypi.org/products/raspberry-
              pi-2-model-b/">
      Product this Web Thing is based on.</a>
  </div>
  <div property="links" typeof="Properties">
    <a property="url" href="properties/">
      Properties of this Web Thing.</a>
  </div>
```

If we can't rely on Google…

- **Using semantic markup is not necessarily an exercise in futility, even if Google et al. do not index it properly**
- **It can be used in more specific contexts**
  - writing a general browser side JavaScript library to extract information about Things and provide fancy controls and visualisations
  - targeted towards more specialised search engines that *do* index the semantics

Summary

- **Discovery poses a number of challenges for a Web based architecture for the Internet of Things**
  - it is not well supported from an architectural point of view (wrong level in the stack)
  - the mechanisms in place for Internet scale discovery (search engines) cater to ordinary Web pages rather than Things
  - Semantic annotations may help, but requires adoption by The Powers That Be, and the track record is not exactly promising (which is, admittedly, a Catch-22)
- **Is it, *really*, such a problem?**
  - what are the use cases where you would *want* arbitrary users/systems from across the Internet to connect to your Things?
  - *local* discovery using a zero-configuration system is important, but Internet scale?
  - why not write your own WoT search engine, and make Google point to it instead?