General announcement

- **Next Tuesday: Guest lecturer, recording unlikely, and probably highly relevant for projects & exam**
  - turn up in person!

Discovery & Security for the Web of Things

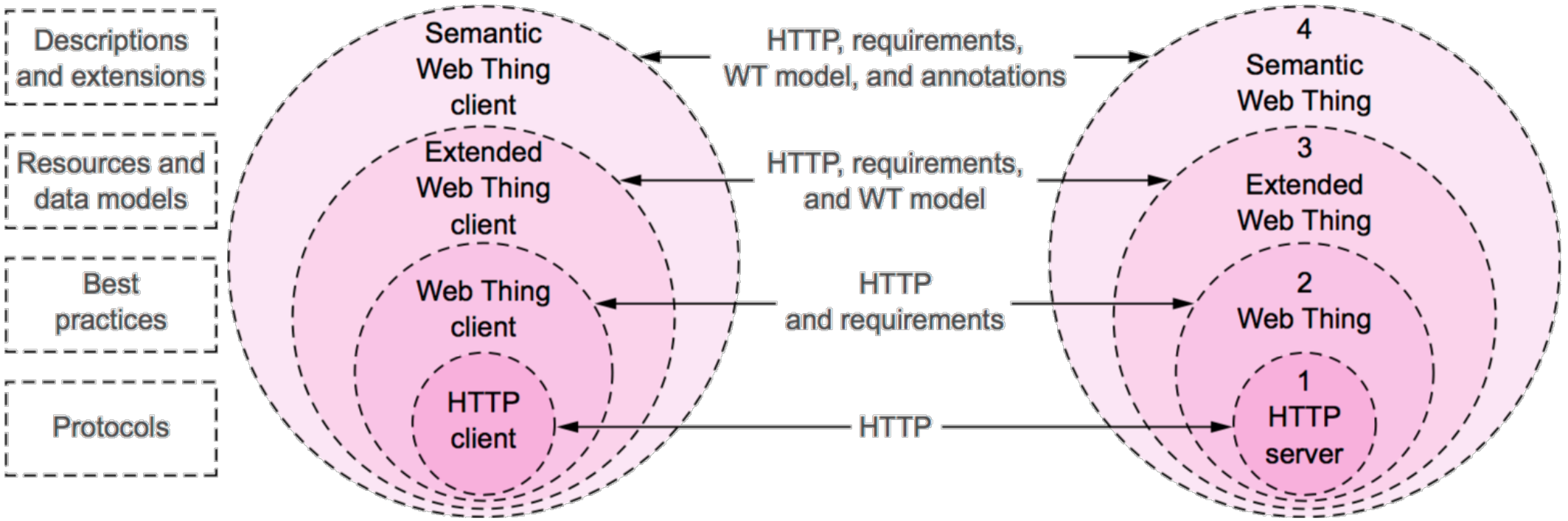
Niels Olof Bouvin

The Challenge of Interoperability

JOKE!

- **What if… there was a *4th* milestone?**
  - "integrate all other groups' Things into your storage system". . . due Monday!
- **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**
  - sufficiently well to derive 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
- The state of IoT security
- Securing and sharing the 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, though that frankly is a bit of a hack
- **So, how might we discover which Things are present?**

Discovery services for IP

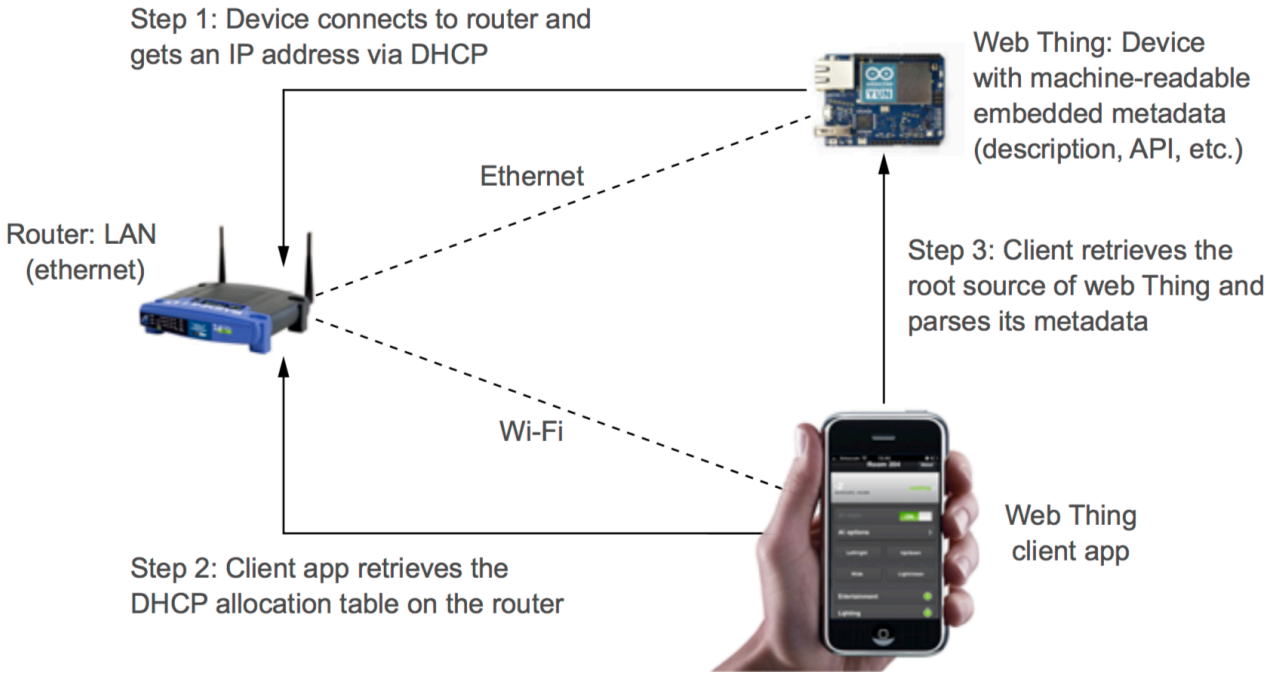
- **Discovering new devices on the local network is not a new problem. Often referred to as zero-configuration**
  - DHCP
  - UPnP, DLNA
  - mDNS, Bonjour (macOS), Avahi (Linux)
- **Usually handled through a broadcast across the local network, where devices can announce or identify themselves**
  - to 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**
- **The state of IoT security**
- **Securing and sharing the 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 those pages
  - but what about tiny Things? What about JSON and other unHTML formats?
- **Plus, a well-understood structure will ease coding**

IETF RFC5988

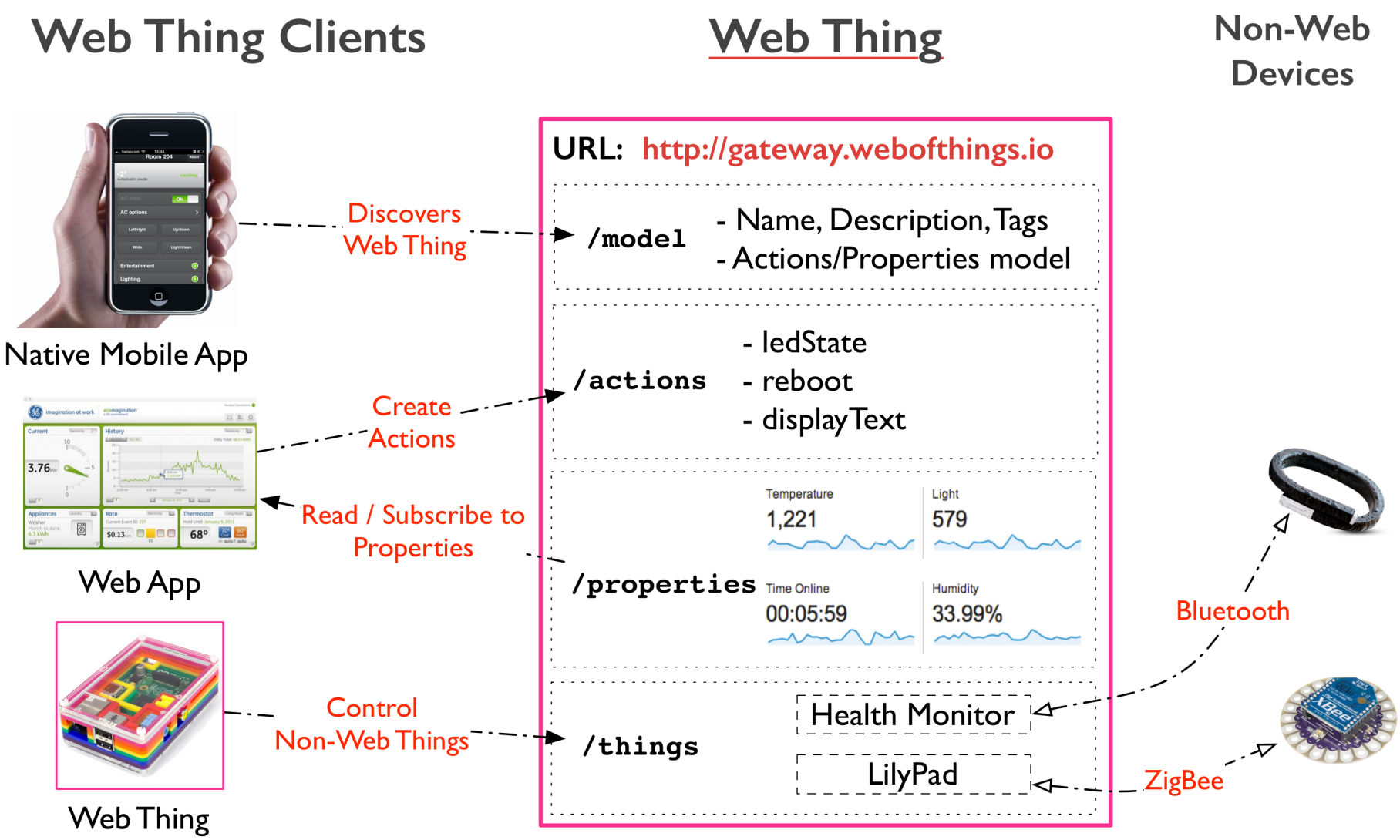
```
–> Request
HTTP 1.1 GET /
Host: MyLittleThingie.io
Accept: application/json

<– 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
  - and gateways controlling many other things

The Web Thing Model



Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

- **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

```
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
  - still no verbs—just nouns

```
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

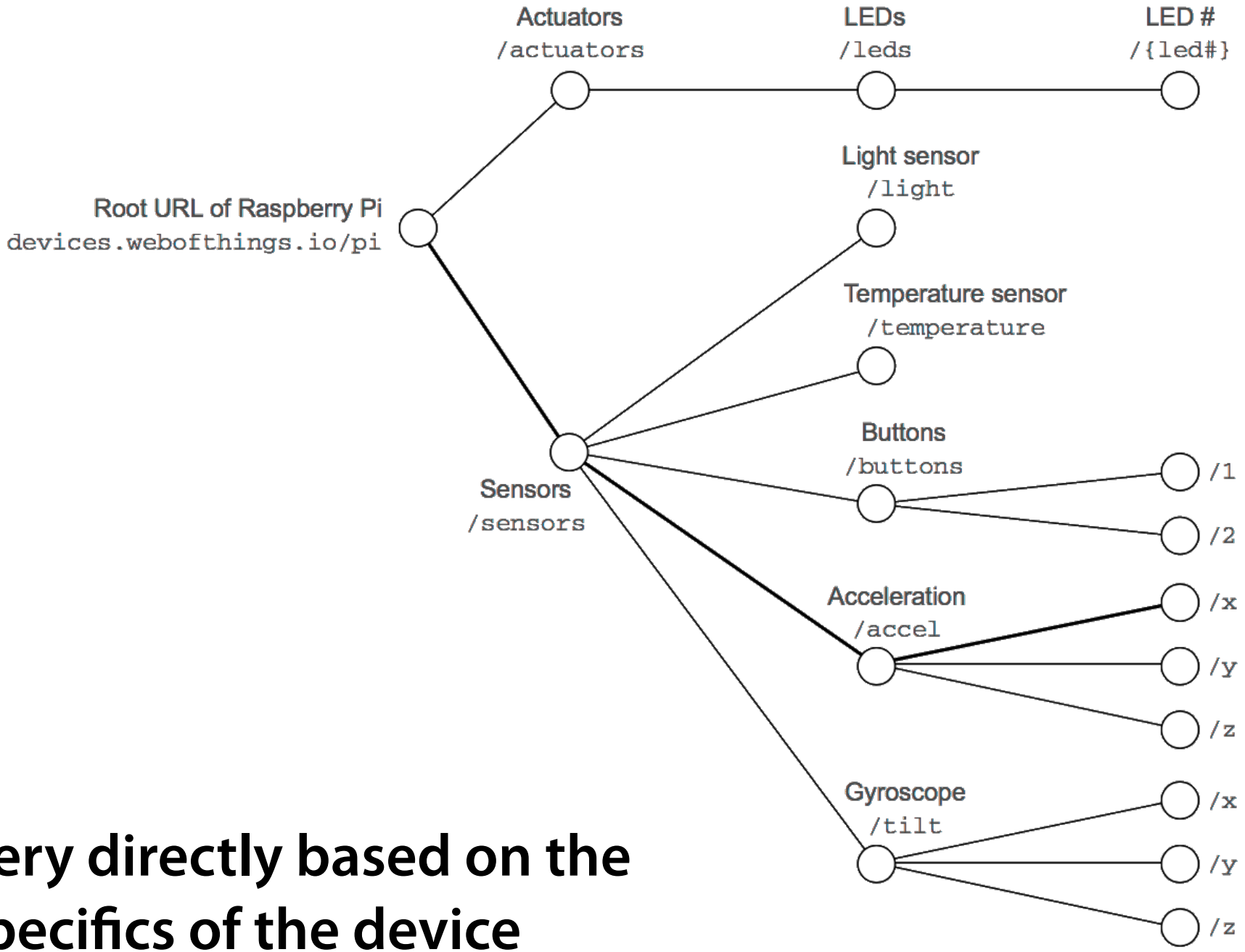
```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Link: <model.webofthings.io>; rel="type"
...

"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 lightbulb
- **These 'sub'-Things can be contacted through the Thing**

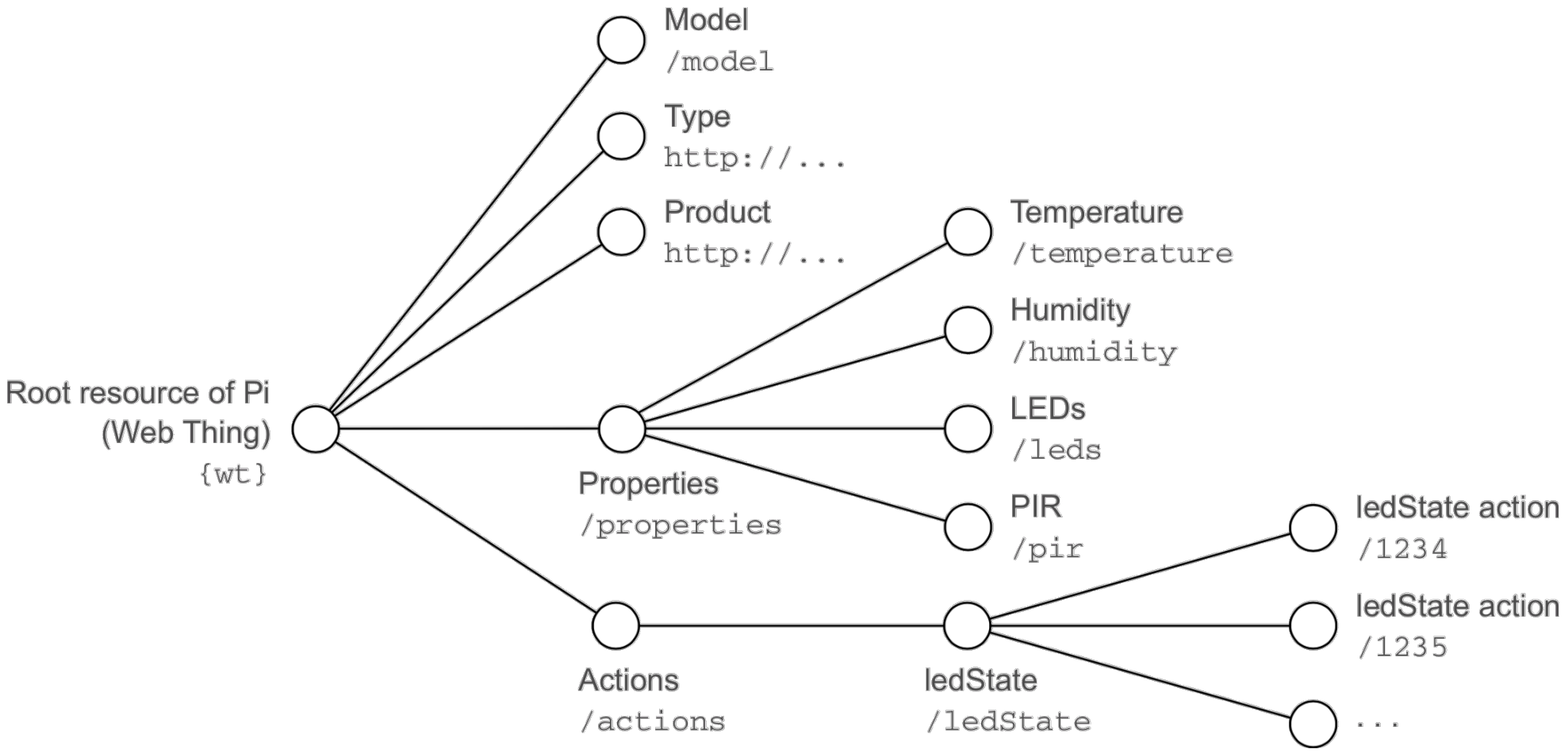
```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Link: <model.webofthings.io/things>; rel="meta"

{
    {
   "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



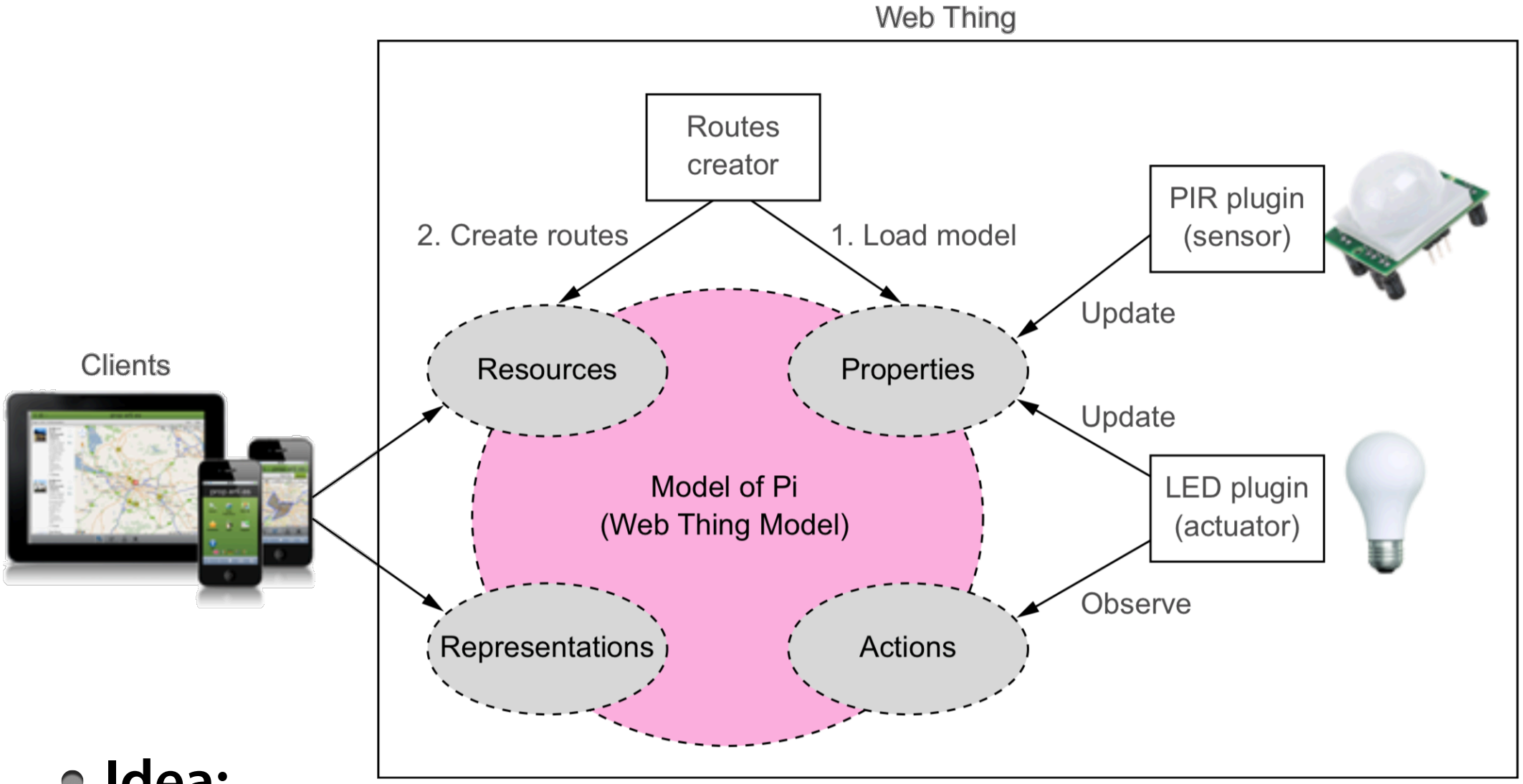
- Very directly based on the specifics of the device

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 state of IoT security**
- **Securing and sharing the 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?
  - why should we trust their metadata more 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—for two decades, at least
- **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. do 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 to "soon" index 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 (SEO is the attempt to reverse-engineer these secrets)
  - 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?

Overview

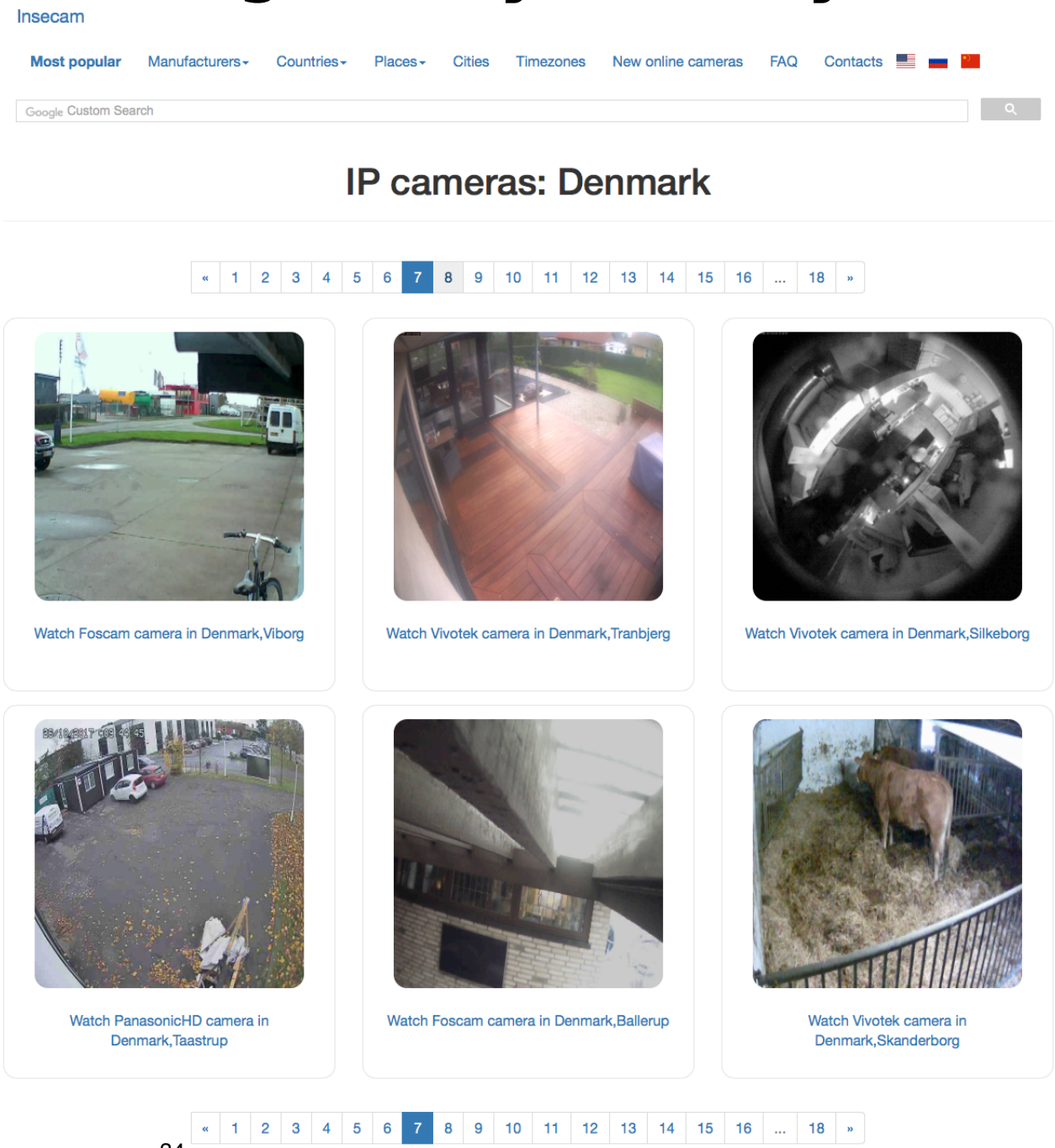
- Discovering Things
- Towards a general model for Things
- The Semantic Web of Things
- **The state of IoT security**
- **Securing and sharing the Web of Things**

The Internet of Things

- **Myriad interconnected devices, reporting and controlling**
  - many different suppliers
  - many different architectures and systems
  - many different use situations ranging from trivial to absolutely crucial
  - many different actors and agendas
- **What could *possibly* go wrong?**
- **Very early days, yet things are not well**

http://www.insecam.org/en/bycountry/DK/

- Maybe, just *maybe*, you should set a password?



Types of attacks

- **Denial of Service**
  - removing ability to use a device or a service
  - annoying, if I can't use my IoT toaster; catastrophic, if national power grid is down
- **Surveillance**
  - by the state, by commercial interests, by criminals
- **Intrusion**
  - a root kit on a smart device inside an installation could potentially compromise all devices within network reach

Fear the IoT_Reaper

- **One example of a Botnet attacking surveillance cameras, home routers, NAS boxes, etc, last year**
  - devices from D-Link, TP-Link, Avtech, Netgear, MikroTik, Linksys, Synology, and others
- **Estimates put the number of infected machines around 1-2 mill.**

How did we get here?

- **Many IoT devices have *very* poor security**
  - unencrypted traffic
  - firmware not being patched—either by the manufactor or the owner
  - best security practices not followed
- **They are in homes and companies—*inside* the firewall**
  - and can thus act as trojan horses or vectors for attacks
  - as well as surveillance and industrial espionage
  - (this is why *all* communication inside and outside your network should be encrypted)
- **This is not bad. This is really, *really* bad**

Network level security

- **Challenge: Heterogeneity**
  - strong cryptography may be straightforward to implement on ordinary computers, but what about much more constrained devices?
  - public key infrastructure may be difficult to handle in a large IoT setting
  - gateways can handle part of the burden
- **Centralistic solution simplest, but also a single of point of failure**
  - competent actors will act responsible, but that still leaves the rest. . .
  - and what happens if the company goes under, and the domain is bought by hackers?

Privacy

- **A user's data should belong to the user**
  - unless this can be ensured, the IoT can become the perfect surveillance infrastructure
- **Centralistic solutions easier to exploit**
  - how can the user ensure proper treatment of collected data?
- **Distributed solutions keep data closer to the user (and their control)**
  - but leaves more points to attack
- **EU's General Data Protection Regulation (GDPR) helps**
  - but is also quite complicated and comprehensive, yet it is a start

Identity

- **IoT objects must have identities that can be found and authenticated by other services**
  - identities can be fixed (5794-118), or fluid (lecture hall for P2P/IoT course)
  - identities can be revealed or hidden (behind authenticated third parties)
- **Typical much easier to implement in a centralised system—many challenges remain in a distributed system with ad-hoc connections**

Trust

- **How can trust be built?**
- **One thing is the negotiation between devices**
  - based on authentication, negotiation, and observation
- **Another is the trust of users in the IoT**
  - transparency
  - control

Fault tolerance

- **Things *will* go wrong**
- **The system must cope**
  - identify errors and failing sensors
  - choose alternative sensors or services
- **Sometimes systems will come under attack**
  - identify compromised systems
  - route around damage

Summary

- Security and privacy are major requirements for a successful IoT
- So far, there have been plenty of examples of early IoT systems susceptible to attacks
- Clearly, this will have to change
- Industry standards and/or government regulations
- Who owns the data?
  - the generator of the data?
  - the provider of the service?

Overview

- Discovering Things
- Towards a general model for Things
- The Semantic Web of Things
- The state of IoT security
- **Securing and sharing the Web of Things**

The elements of a secure Thing

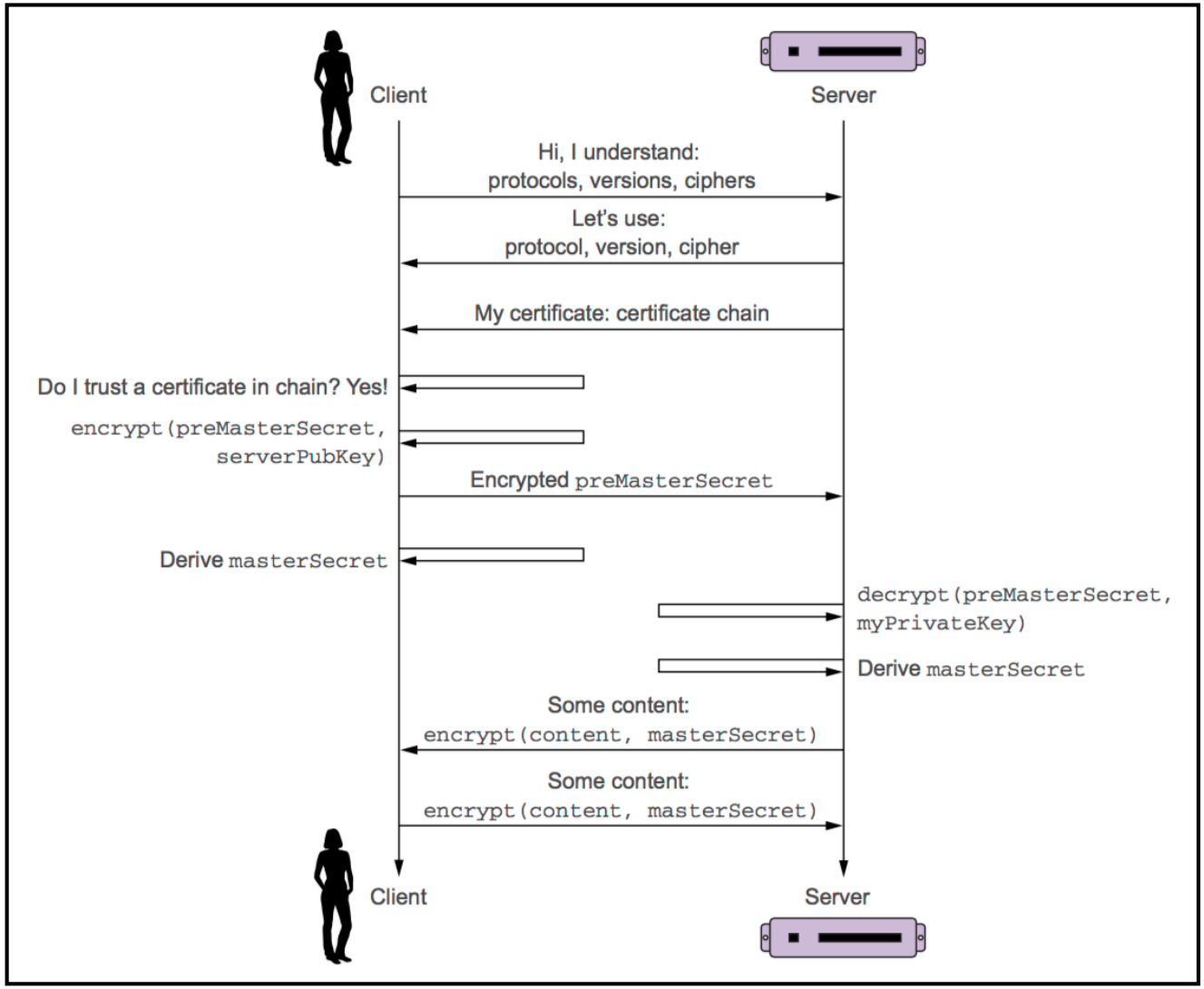
- Encrypted communications
- Authenticated servers
- Authenticated clients
- Secure access control
- Secure software updates

Encrypted communications

- The basic requirement
- Provided that encryption keys are exchanged securely, this should ensure no eavesdropping
  - assuming best practices are followed, of course
- Asymmetric and symmetric encryption combined

Authenticated server

- We need to know the Thing is the *actual* Thing
  - HTTPS and TLS to the rescue
- Keys can be generated locally, and HTTPS support is in Node.js
- https://letsencrypt.org/ provides free certificates



Authenticated client/user

- **Some IoT devices may not need user authentication**
  - weather stations, public sensors, . . .
- **Others certainly require it**
  - cameras, anything with an actuator, anything privacy sensitive

Letting users in

- **Simplest approach: create user profiles with**
  - user names
  - passwords
  - privileges
  - etc
- **Require authentication over secure connection before access is granted**
- **Access, once granted, handled through a token**
  - generated by the server
  - exchanged in headers between client and server

OAuth

- Rather than having users remember yet another password (and having to store that securely), let users connect using preexisting identities
- The user authenticates themselves to a known service, and that service then authorises access to their API for that user from your server
- Not a perfect system—users have been fooled by phishing attacks with sites purporting to be, e.g., Google Docs or nemID requesting authorisation
  - though this is no different than ordinary phishing attacks

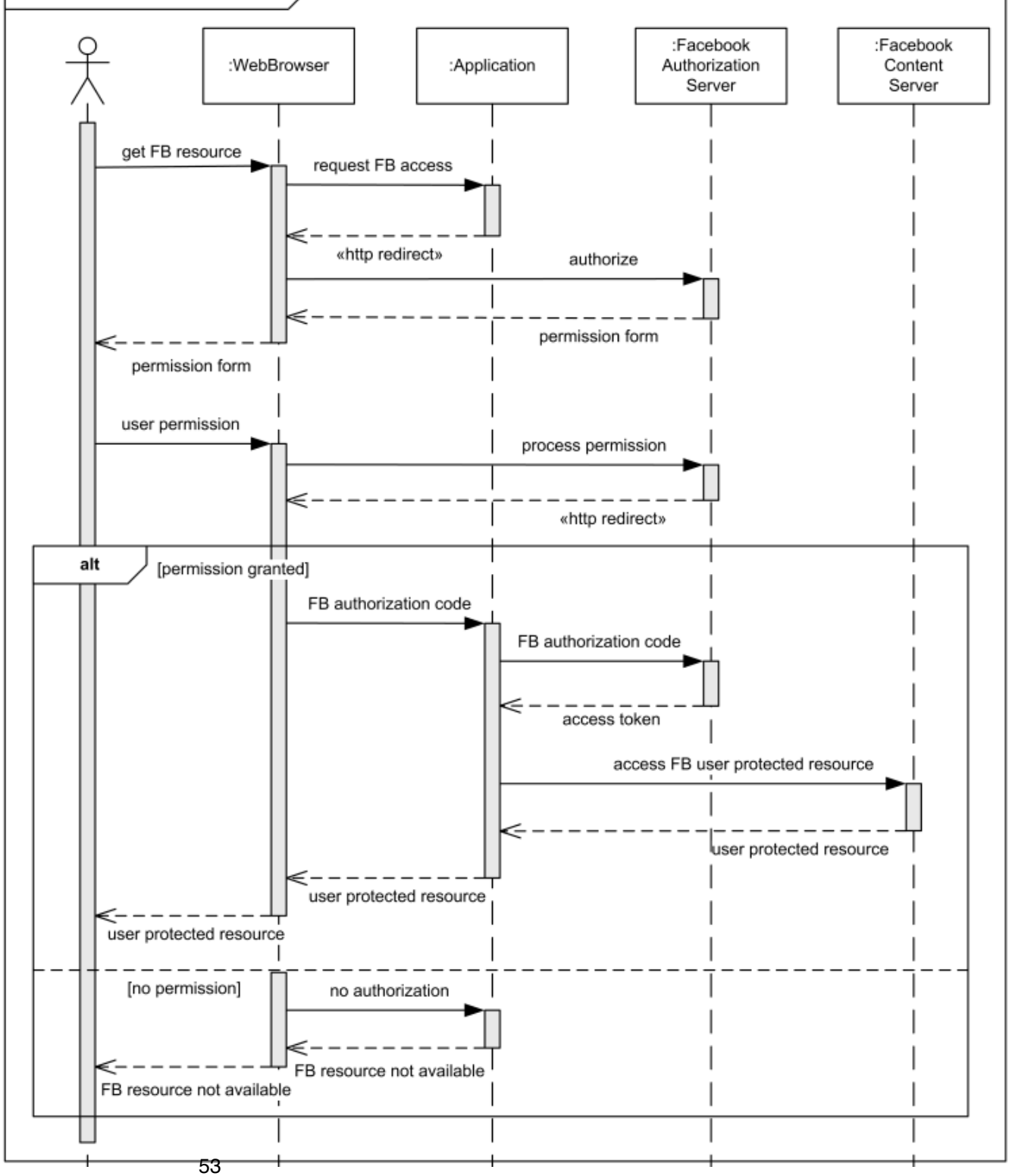
Roles in OAuth

- **The application**
  - the application that needs access to some part of the user's account for, e.g., identity
- **The resource server**
  - provides the API for accessing the user's account
- **The authorisation server**
  - handles the interaction with the user granting/denying access (i.e., login to Twitter)
- **The resource owner**
  - the user, who is granting/denying access to part of their account at the resource server

Requisites

- **The application must be registered with the authorisation server, which provides**
  - client id (this can be public information)
  - client secret (this cannot)
- **The application must provide an redirection URL**
  - which must be secure, e.g., HTTPS

- The interaction between a user, an application, and Facebook authorisation & content servers
- Tokens exchanged through redirects
- Given the access token, the application can access the Facebook API

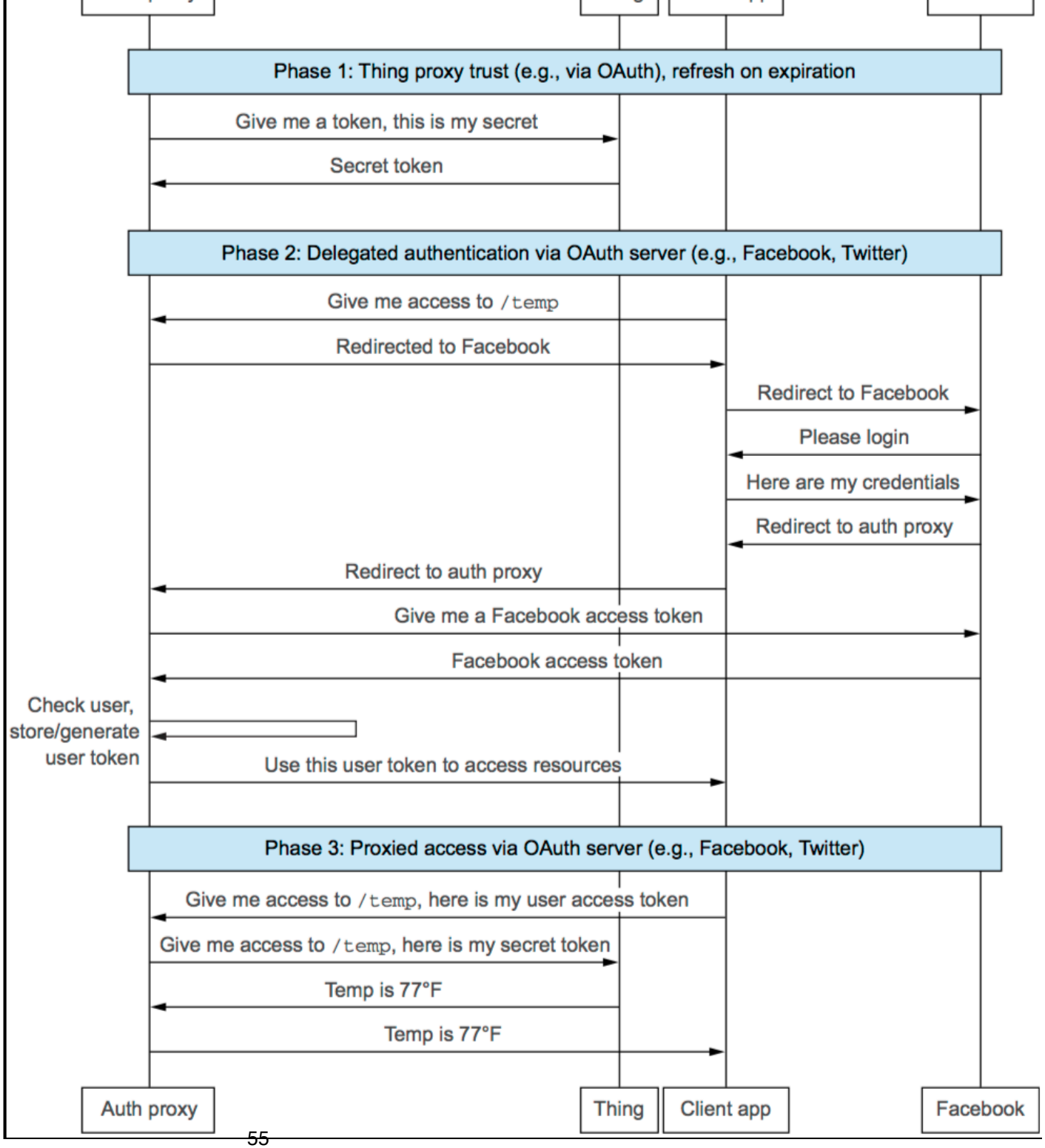


The Social Web of Things

- **Using OAuth helps with authorisation and identities of users**
- **Having each and every Thing maintain lists of users is a bit cumbersome though…**
- **An Authentication Proxy could handle this interaction**
  - registering all the Things
  - handling access to the authorisation servers through OAuth
- **One Authentication Proxy for, e.g., a building, a company, or a home**

SWoT flow

- Using the token generated, the user can then access the protected resources
  - in *Fahrenheit?*



User identity is not enough

- User roles must also be defined, also known as Access Control Lists
- Some users can be administrators, others cannot or should not

So… what if there is a bug?

- **Software contains bugs, and so will IoT devices**
- **Providing a secure mechanism to push software updates out to devices becomes crucial**
  - if the software is not updated, security holes may not be patched, and new features cannot be added
  - if the update mechanism is compromised, the device can be loaded with malware by criminals
  - one solution is an App Store with automatic downloads
- **resin.io provides such a service**
  - based on git, node.js and/or Docker
  - and it's free, if you have no more than 10 devices

Security is tricky

- **Security is not an end goal—it is an ongoing process**
  - all software contains bugs, including security software, so we must adapt over time
  - the web stack is the most used in the world, so its security will receive much scrutiny
- **Many IoT devices provide terrible security, which puts only the devices and their users at risk, but the entire Internet, if these devices are weaponised into botnets**
- **Therefore: do better, be careful, follow best practices**