ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Web of Things Application Architecture - Integrating the Real-World into the Web

Dominique Guinard
Ph.D. Defense, ETH Zurich

Motivation

Why should we bring the Web and real-world devices together?

[flickr.com/photos/moragcasey]

An Increasing Number of Connected Smart Things…

From machines & home appliances…

to sensor networks…

and tagged objects.

A very large ecosystem of smart things, complex application development

Need for a Common Internet of Things Application Architecture

- Application development with smart things:
  - Requires expert knowledge:
    - Hardware/software heterogeneity
    - Lack of common application protocols
  - WSN [Mot2011]
    RFID [Sch2008]

- **Hypothesis:** The Web (application archi. of the Internet) can be the application architecture of smart things as well.
- **Research Question:** *«How can the Web be leveraged to ease the development of Internet of Things applications and bring it closer to non-specialists?»*

[Sch2008] Schmitt, P. *Adoption und Diffusion neuer Technologien am Beispiel der Radiofrequenz-Identifikation (RFID)*. PhD Thesis, ETH Zurich.

[Mot2011] Mottola, L., & Picco, G. P. *Programming wireless sensor networks: Fundamental concepts and state of the art*. ACM Comput. Surv.

Contributions

- A Web of Things Application Architecture:
  - Adapt and leverage protocols, services and tools of the Web ecosystem
  - Foster a participatory application development:
    - Easier for specialists
    - Closer to Web developers (Web languages), tech-savvies (mashups) and end-users (browsers)
  - Evaluated for WSN and RFID:
    - Simplifies the development and deployment of applications

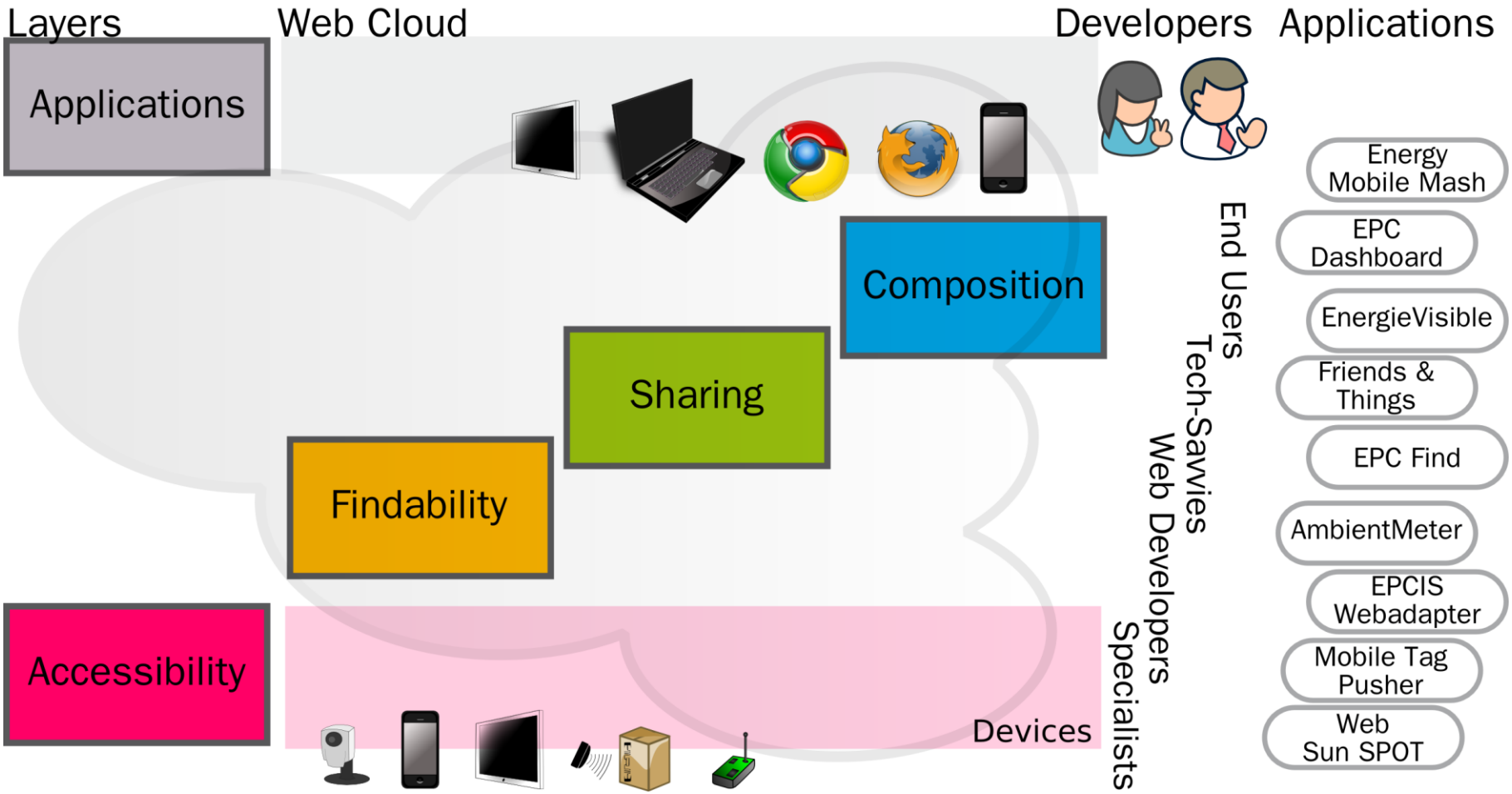
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Web of Things Application Architecture

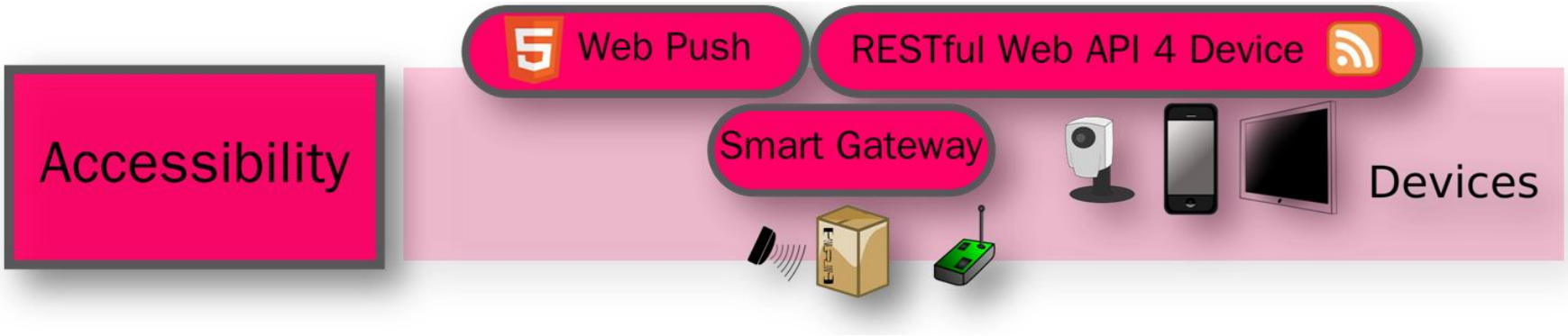
Simplifying Application Development in the Internet of Things

[flickr.com/photos/docman]

Web of Things Application Architecture



Device Accessibility Layer



- *How do we make smart things accessible on the Web?*
- Generic design process[Gui2010] for smart things as Web resources:
  - REST[Fie2000] and Resource Oriented Architectures[Ric2007]

[Fie2000] Fielding, R. (2000). *Architectural styles and the design of network-based software architectures.* PhD Thesis

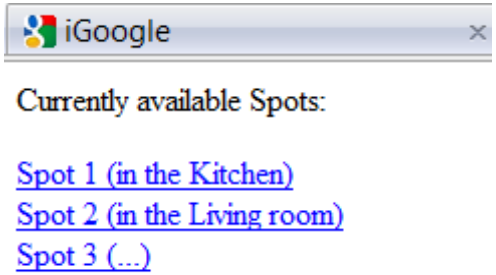
[Ric2007] Richardson, L., & Ruby, S. *RESTful web services*, O'Reilly Media.

[Gui2010] Guinard, D., Trifa, V., Wilde, E. *A Resource Oriented Architecture for the Web of Things*. IoT 2010

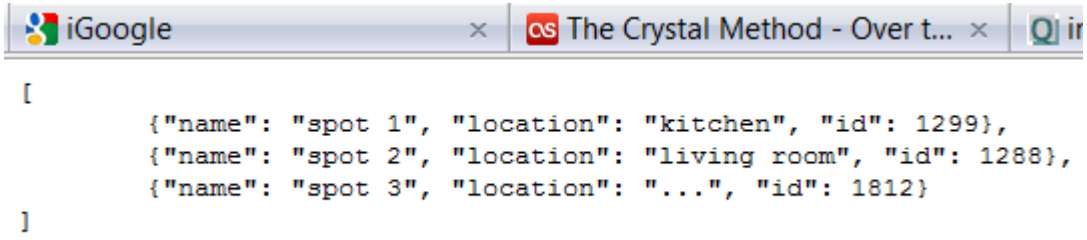
Accessibility

Resource Design → **Representation Design** → **Interface Design** → **Implementation Strategy**

http://<DOMAIN>:<PORT>/genericNodes
/node1/sensors/temperature

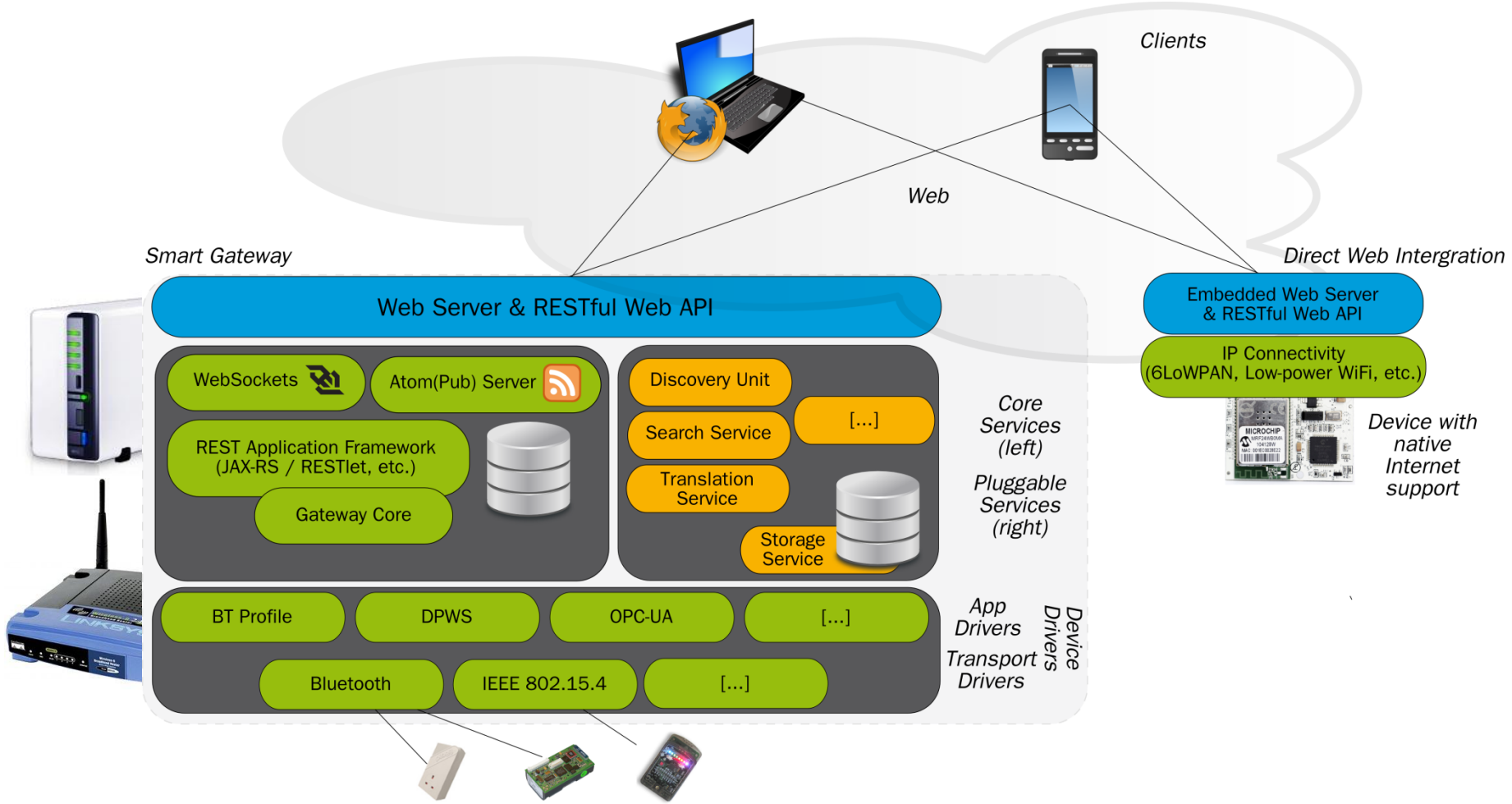


http://<DOMAIN>:<PORT>/genericNodes

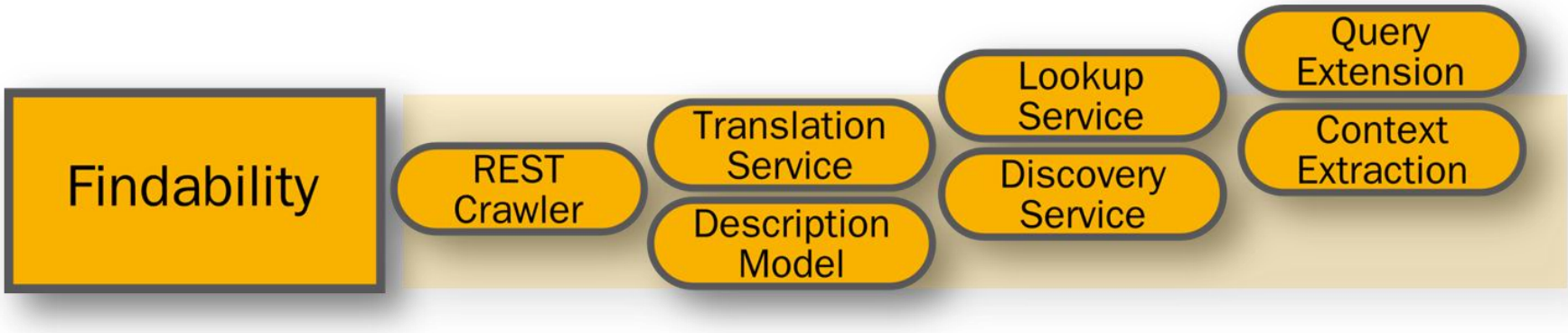


Accessibility

Resource Design → **Representations Design** → **Interface Design** → **Implementation Strategy**



Findability Layer



- *Once smart things are accessible on the Web, how do we enable users to find the right service for their application?*
- Enabling Smart Things to be indexed by search engines (lightweight metadata)[Gui2011]
- Local lookup and discovery infrastructure [Gui2010a,May2011]

[Gui2011] Guinard, D., Trifa, V., Mattern, F., & Wilde, E. *From the Internet of Things to the Web of Things.* Architecting the Internet of Things (pp. 97-129)

[Gui2010a] Guinard, D., et al. (2010). *Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services*. IEEE Transactions on Services Computing

[May2011] Mayer, S., Guinard, D. *An Extensible Discovery Service for Smart Things.* WoT2011

Sharing

Sharing Layer

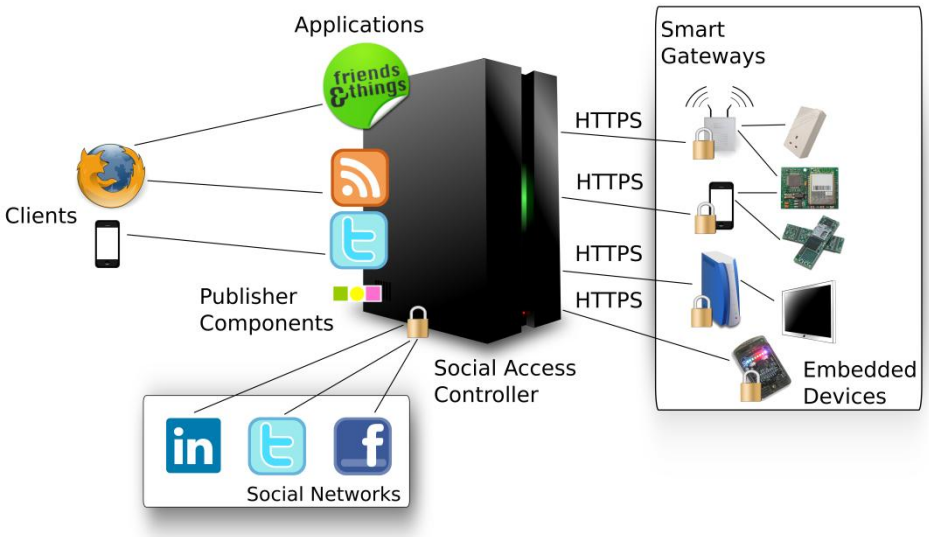
- *Once smart things are accessible and findable on the Web, how do we share them?*
- Social Web of Things [Gui2010b]

[Gui2010b] Guinard, D., Fischer, M., & Trifa, V. *Sharing using social networks in a composable web of things*. WoT 2010

Sharing

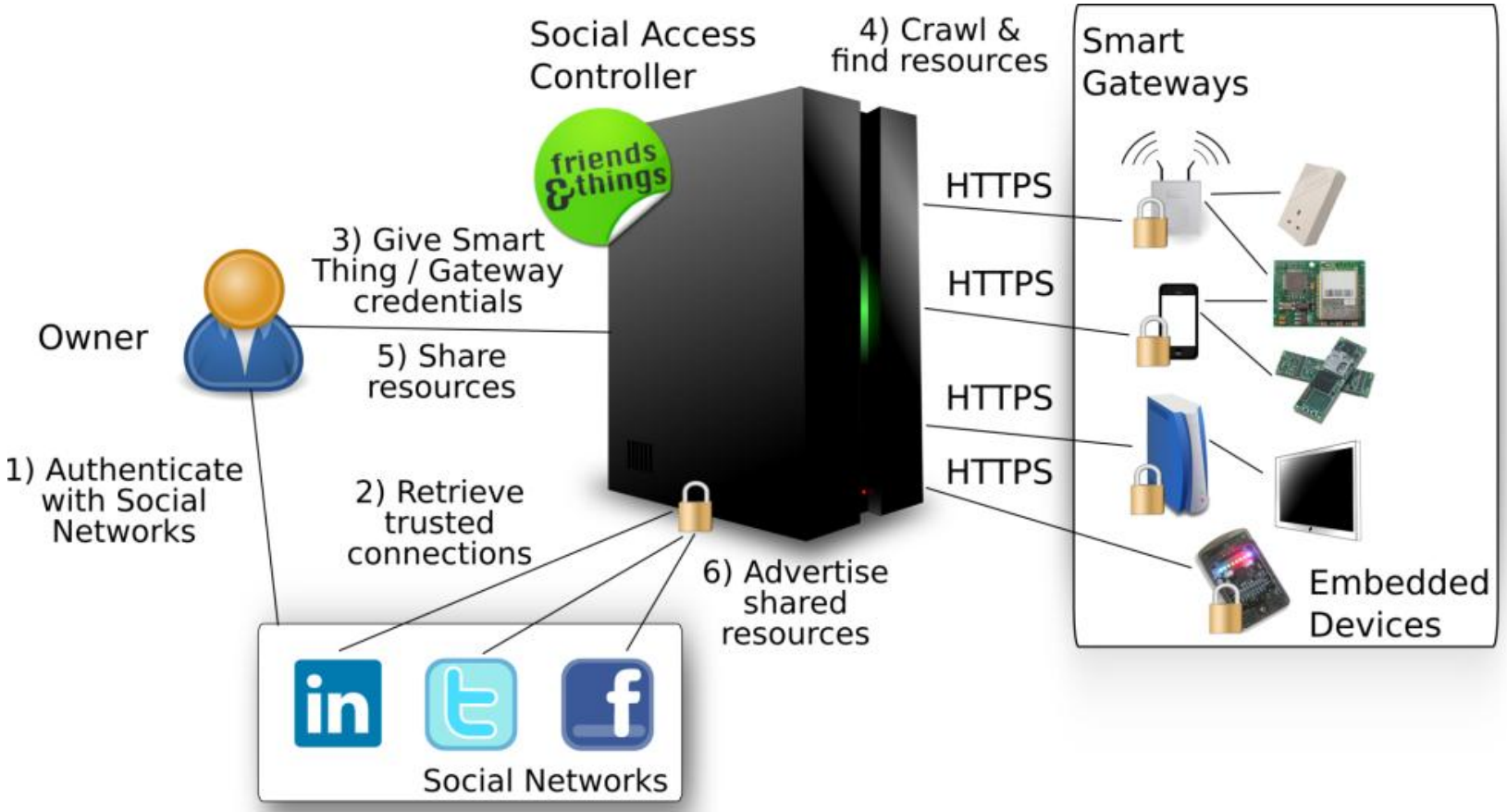
Social Access Controller (SAC)

- Existing systems:
  - Require dedicated access control lists (e.g., HTTP Digest or Basic Authentication)
- Leverage social graphs of social networks:
  - Are walled-gardens [Ber2009]
  - Allow sharing data, not services
- Social Access Controller as proxy between clients and smart things



[Ber2009] Tim Berners-Lee. *Twenty years: Looking forward, looking back.* WWW 2009

Social Access Controller (SAC)



Sharing in Friends and Things

About | Authentication | Resources | Gateways | Shares

Here you can see all Resources that you have shared with your friends or you can share new Resources. For existing Shares, you can display usage statistics in order to see whether it was worth sharing that Resource.

Gateway:
Select a gateway.
vswot.inf.ethz.ch:8081

Social Network:
Select a Social Network to display friends.
Facebook

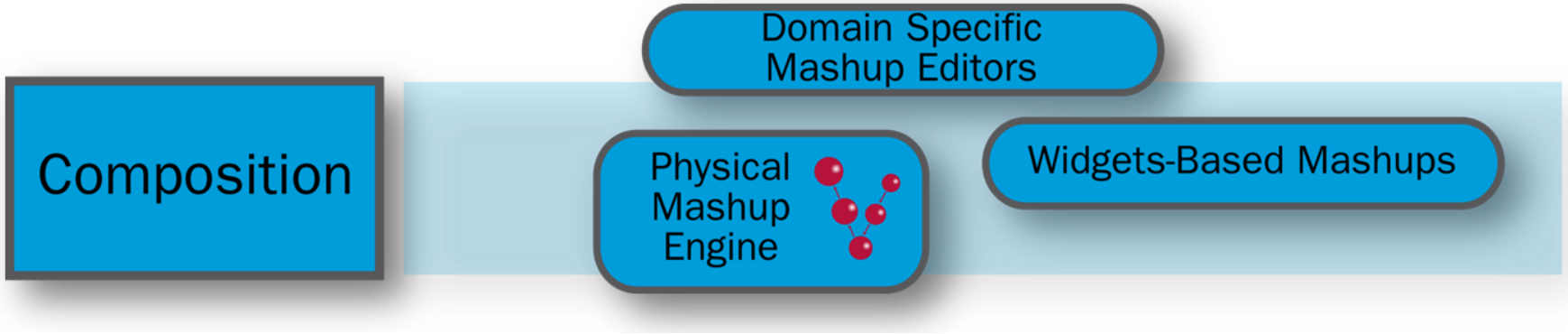
URL:
Select a Resource to be shared. Loading all available resources might take some time, please be patient.
Temperature (/sunspots/Spot1/sensors/temperature)

User:
Select a friend so share a resource with.
Véronique Guinard

- 1 (/sunspots/Spot1/sensors/analogin/1)
- 2 (/sunspots/Spot1/sensors/analogin/2)
- 3 (/sunspots/Spot1/sensors/analogin/3)
- Light (/sunspots/Spot1/sensors/light)
- Switches (/sunspots/Spot1/sensors/switches)
- Switch 1 (/sunspots/Spot1/sensors/switches/1)
- Switch 2 (/sunspots/Spot1/sensors/switches/2)
- Temperature (/sunspots/Spot1/sensors/temperature)
- Tilt (/sunspots/Spot1/sensors/tilt)
- Tilt X Axis (/sunspots/Spot1/sensors/tilt/x)
- Tilt Y Axis (/sunspots/Spot1/sensors/tilt/y)
- Tilt Z Axis (/sunspots/Spot1/sensors/tilt/z)
- Sun SPOT Spot2 (/sunspots/Spot2)
- Actuators (/sunspots/Spot2/actuators)
- LED List (/sunspots/Spot2/actuators/leds)
- LED 0 (/sunspots/Spot2/actuators/leds/led0)
- input (/sunspots/Spot2/actuators/leds/led0/blue)
- input (/sunspots/Spot2/actuators/leds/led0/green)
- input (/sunspots/Spot2/actuators/leds/led0/red)
- input (/sunspots/Spot2/actuators/leds/led0/switch)

http://vswot.inf.ethz.ch:8091
/gateways/vswot.inf.ethz.ch:8081
/resources/sunspots/spot1/sensors/temperature

Composition Layer



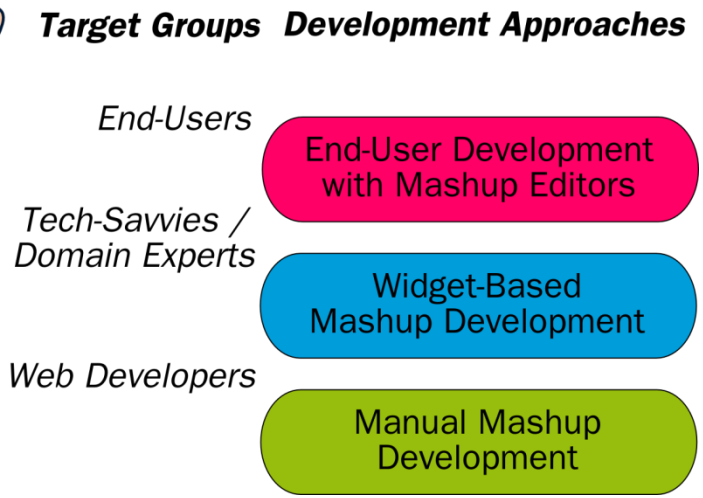
- *Once smart things are accessible, findable, shareable on the Web, how do we enable their easy composition by non-specialists, into new applications?*
- Physical Mashups [Gui2010, Gui2010c]

[Gui2010] Guinard, D., Trifa, V., Wilde, E. *A Resource Oriented Architecture for the Web of Things*. IoT 2010

[Guinard2010c] Guinard, D. *Mashing up your web-enabled home*. ICWE 2010

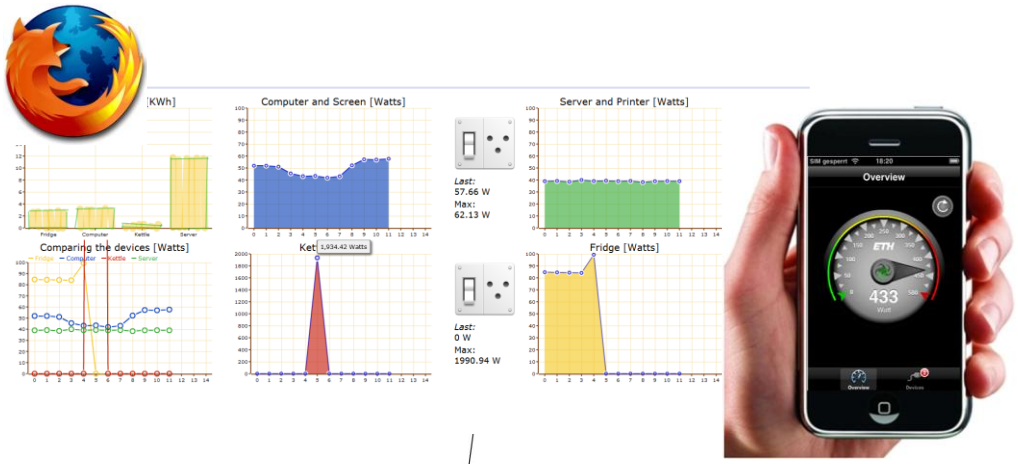
From Web 2.0 Mashups to Physical Mashups

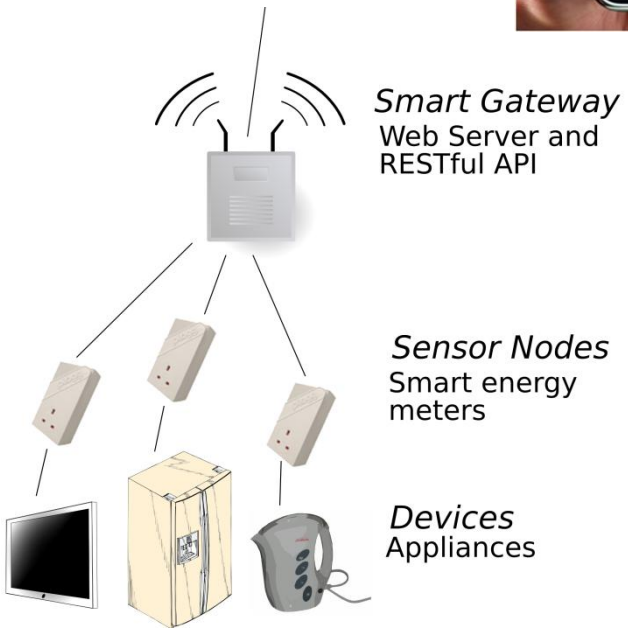
- *Web 2.0 Mashups:*
  - *"Web applications generated by combining […] disparate Web sources […] to create useful new services"* [Yu2008]
  - Ad-hoc applications accessible to a larger public
- Physical Mashups:
  - Composite Web applications involving smart things and virtual Web services
  - Three development approaches



[Yu2008] Yu, J., Benatallah, B., Casati, F., & Daniel, F. *Understanding Mashup Development*. IEEE Internet Computing

Energie Visible: An Energy-Aware Mashup



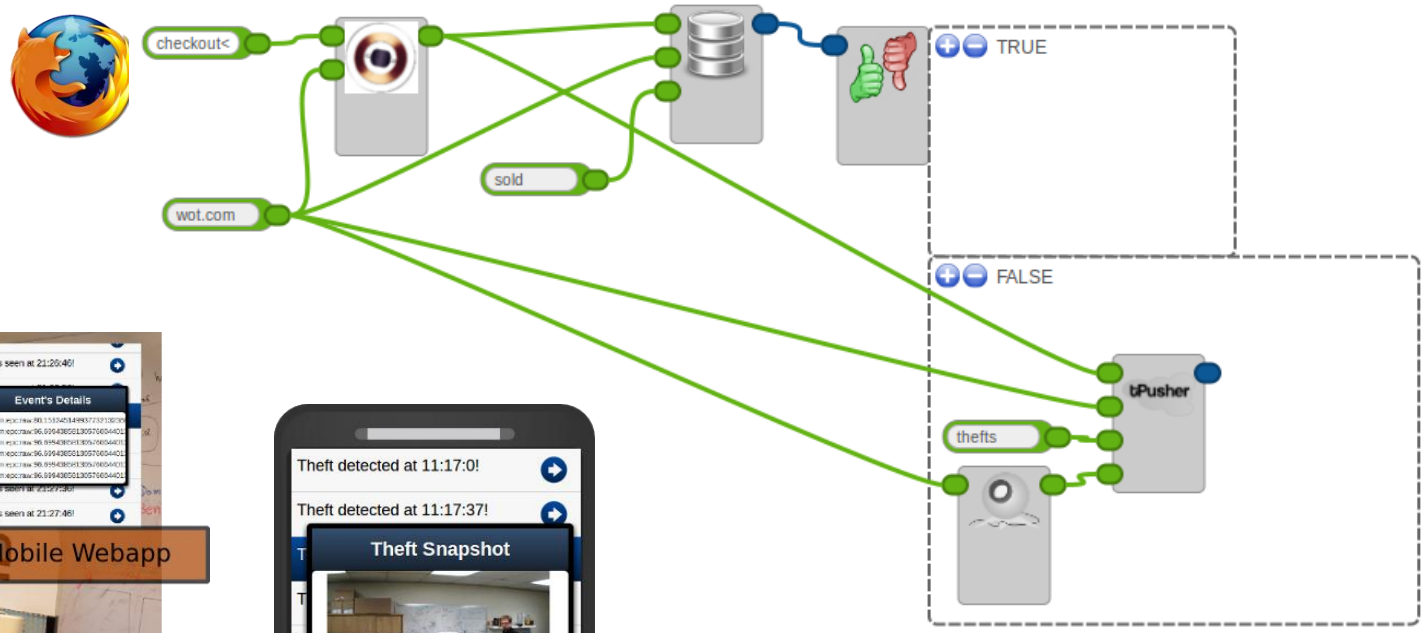


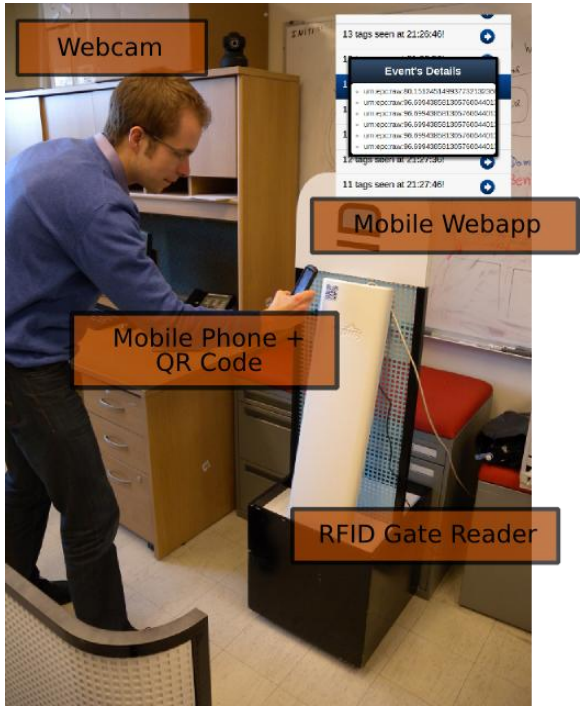
- Developers:
  - Smart Meters as an RESTful Web API:
  - Mashup with any language supporting HTTP
- Users:
  - Used by several families around the world (Energie Visible)

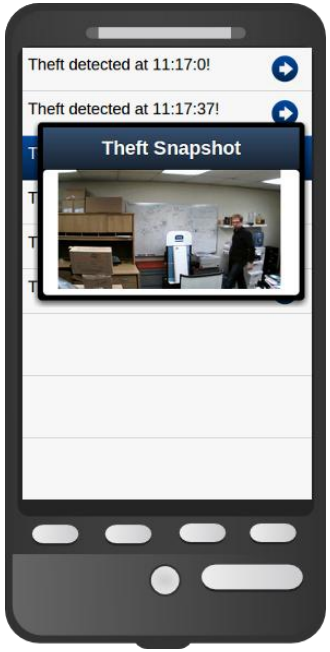
EPC Mashup Dashboard: RFID Business Intelligence

- Developers:
  - RFID Readers & Data in a black-board approach
  - Wizard-based creation of Widgets
  - Merging Web data and real-world RFID data
- Users:
  - Simple Web page providing real-time business intelligence
  - Deployed at the SAP future store

Electronic Article Surveillance as a Physical Mashup







[Gui2010d] Guinard, D., Floerkemeier, C., & Sarma, S. *Cloud Computing, REST and Mashups to Simplify RFID Applications,* WoT 2011

[Naef2009] Naef, L. *ClickScript a visual programming language in the browser*. Master Thesis, ETH Zurich

Selected Evaluations

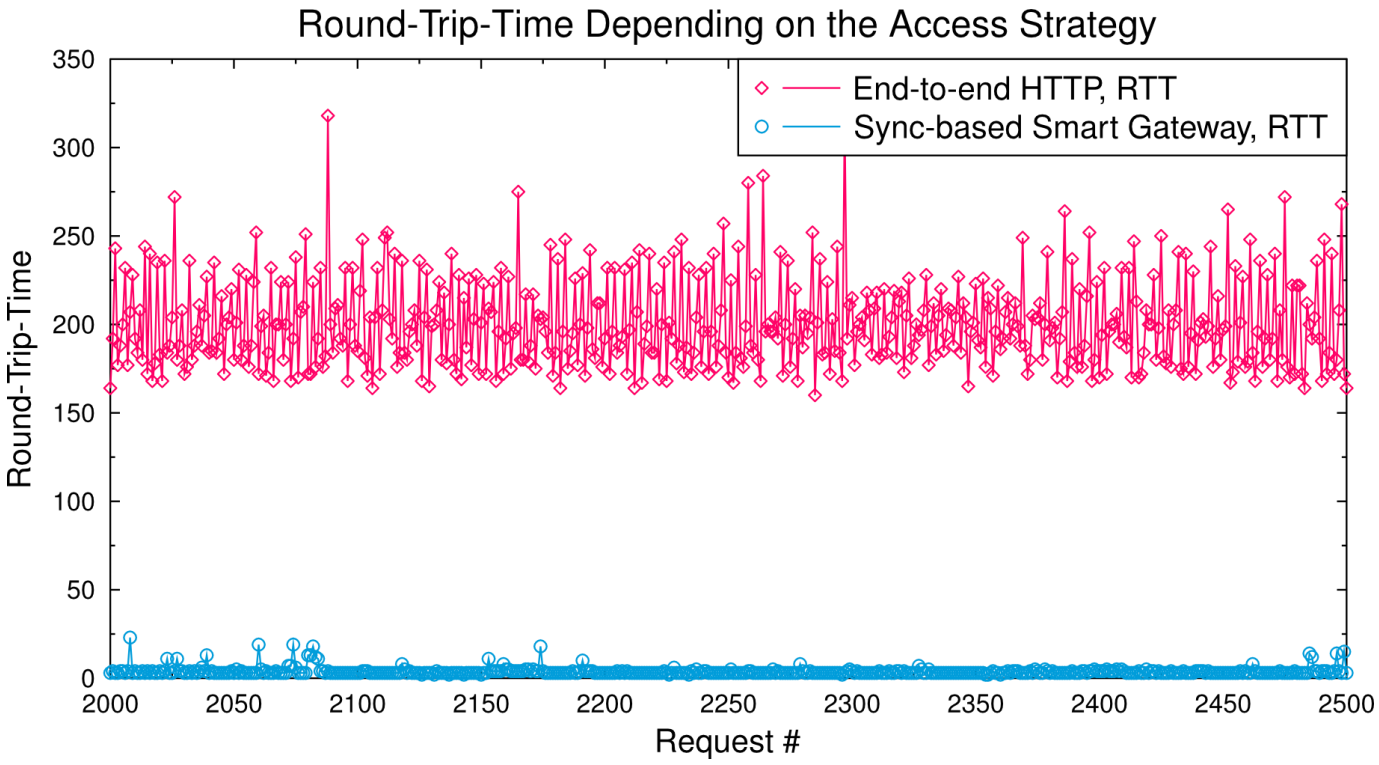
Smart Gateways, Social Access Control & Developers' Experiences

[flickr.com/photos/myfwc]

Accessibility

Smart Gateway vs Direct Web Integration

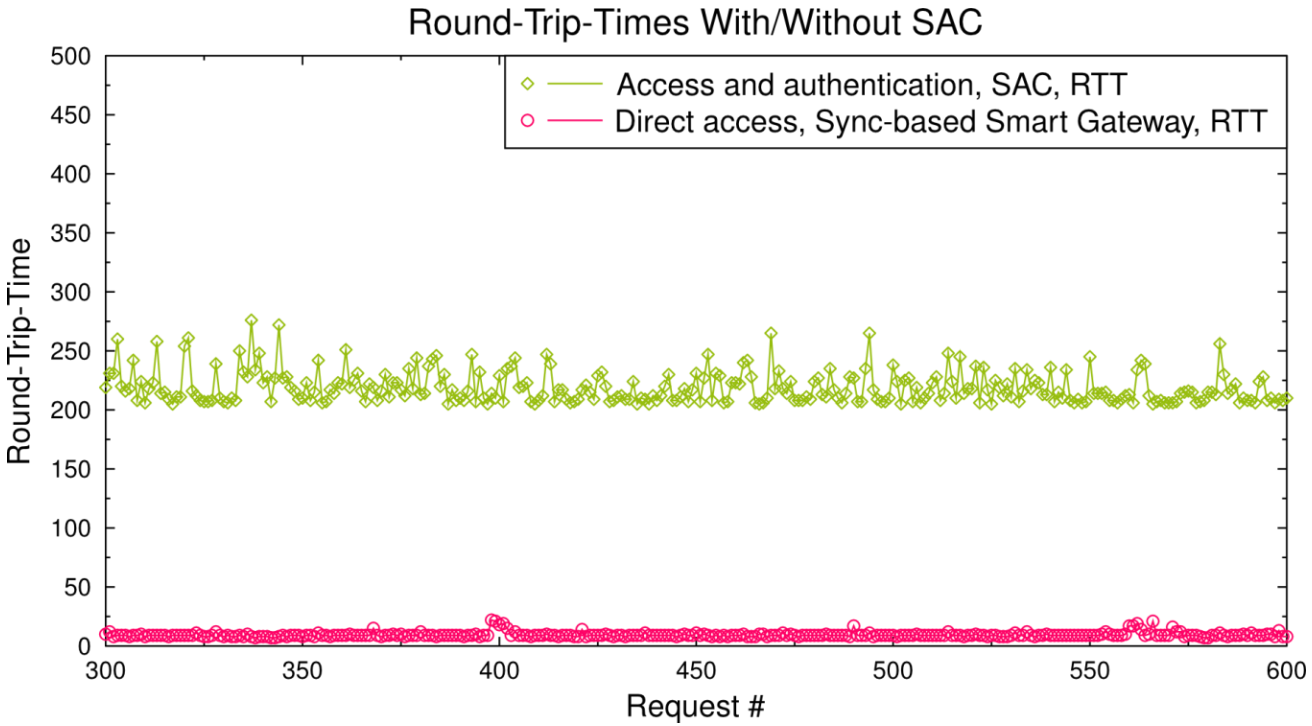
- End-to-end HTTP
  - 7000 (sequential) requests:
    - Avg.: 205 ms, SD: 127.8 ms
    - Min.: 96 ms, Max.: 8.5 sec
- Smart Gateway:
  - 7000 (sequential) requests:
    - Avg.: 4.2 ms, SD: 3.7 ms
    - Min.: 2 ms, Max.: 111 sec



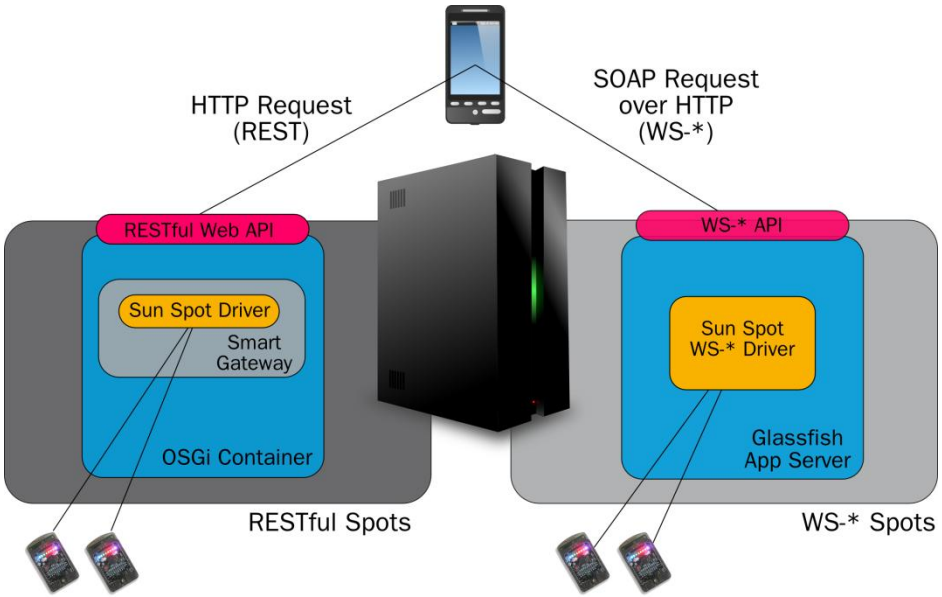
Sharing

Using Social Access Control

- 1000 requests on a Sun SPOT (Smart Gateway):
- Direct Access:
  - Avg.: 9 ms, SD: 2 ms
  - Max.: 40 ms, Min.: 6 ms
- Through SAC (Facebook), RTT:
  - Avg.: 218 ms, SD: 24 ms
  - Min: 204 ms, Max: 830 ms
  - Most of the RTT (140ms) due to social network login => caching



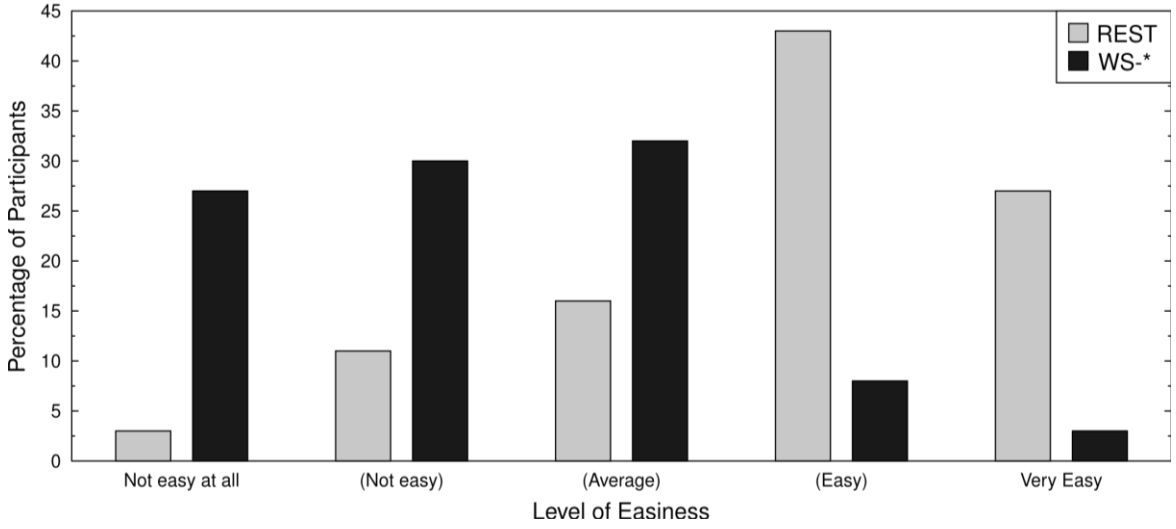
Ease of Use? Assessing the Developers' Experience

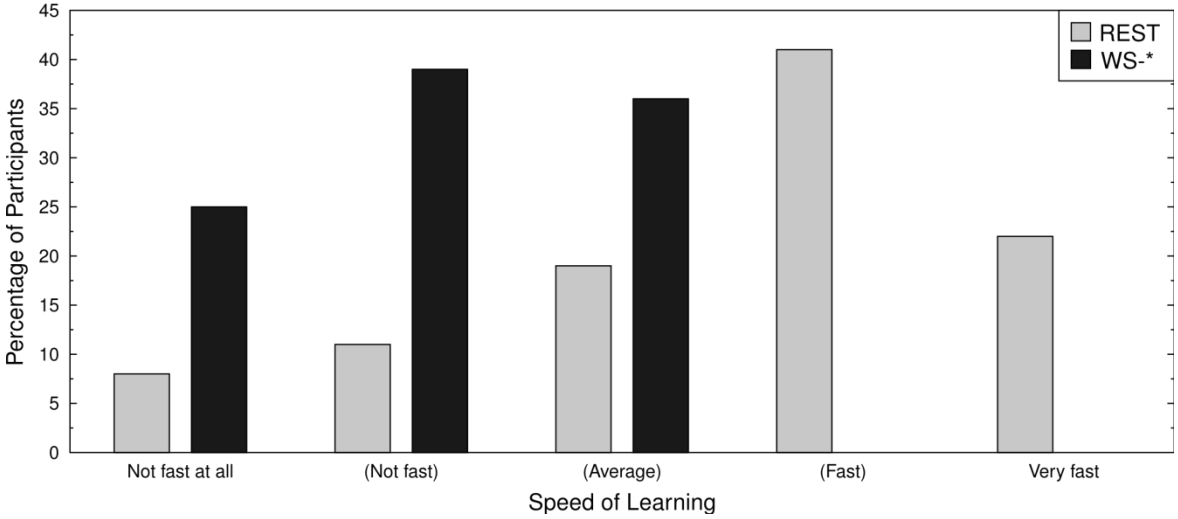


- WS-* (WSDL, SOAP, etc.) as one of the most comprehensive alternatives (DPWS, DNLA, etc.)
- Performances were compared[Yaz2009], not ease of use
- Study with 69 computer science students
- 2 applications:
  1. Android phone accessing a Sun SPOT featuring a WS-* API
  2. Android phone accessing a Sun SPOT featuring a RESTful Web API

[Yaz2009] Yazar, D., & Dunkels, A. *Efficient application integration in IP-based sensor networks*. BuildSys 2009

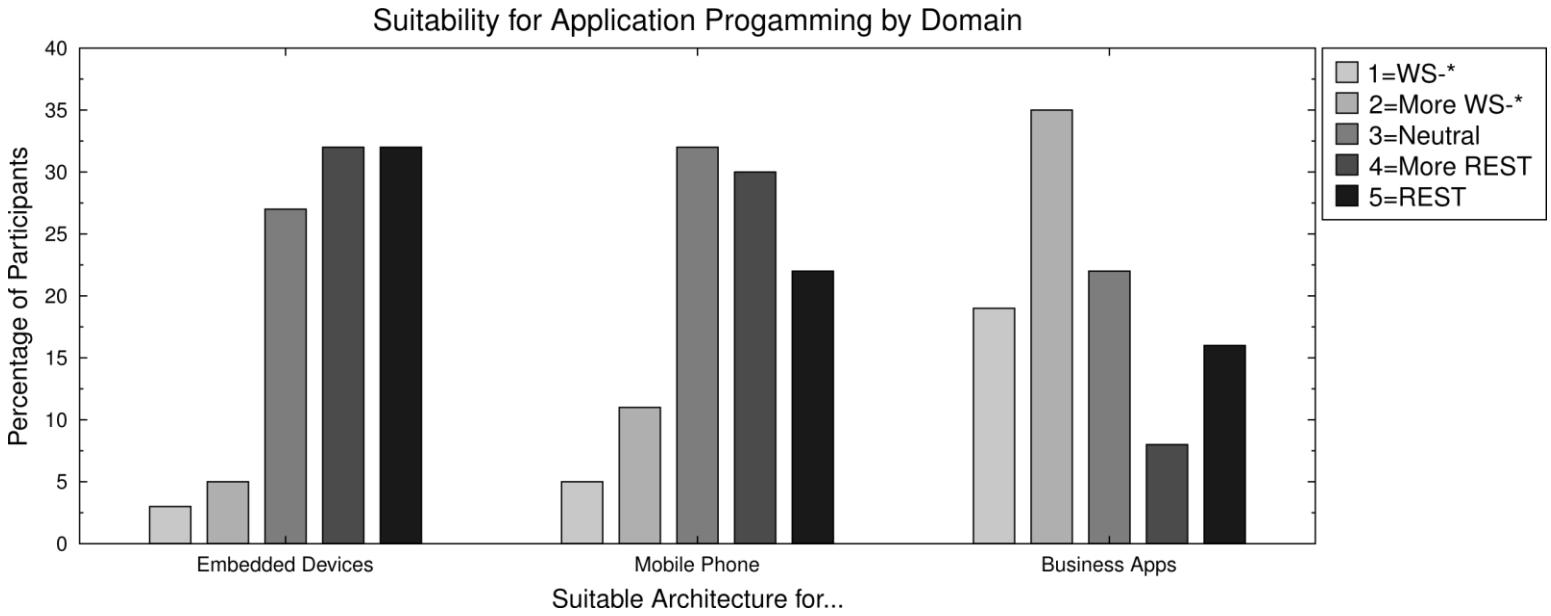
Ease of Learning





- RESTful Web API:
  - 70% easy to very easy to learn
  - 63% fast to very fast to learn

Suitability by Use-Case and Guidelines



- REST and Web more adapted to foster adoption by non-specialists
- WS-* more adapted for high QoS/security requirements

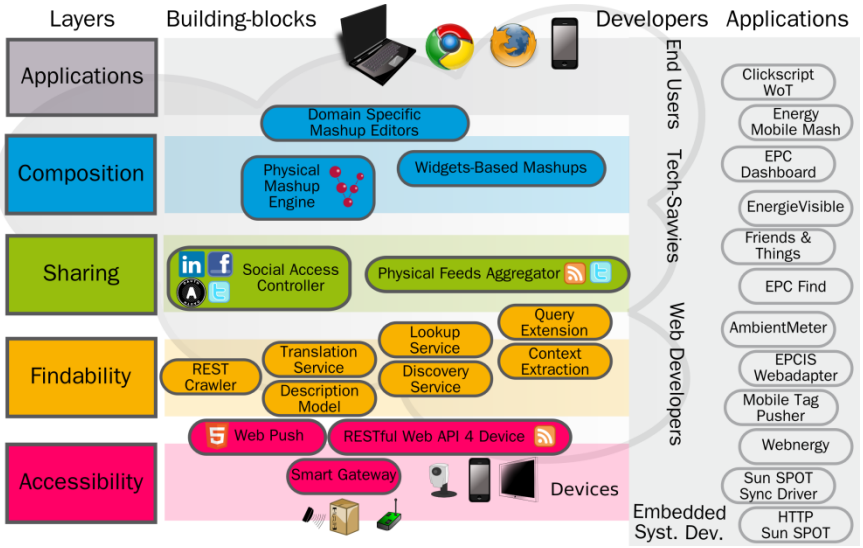
Conclusions, Limitations and Outlook

What did we contribute? What are the current limitations?

[flickr.com/photos/brapke]

Contributions & Learnings

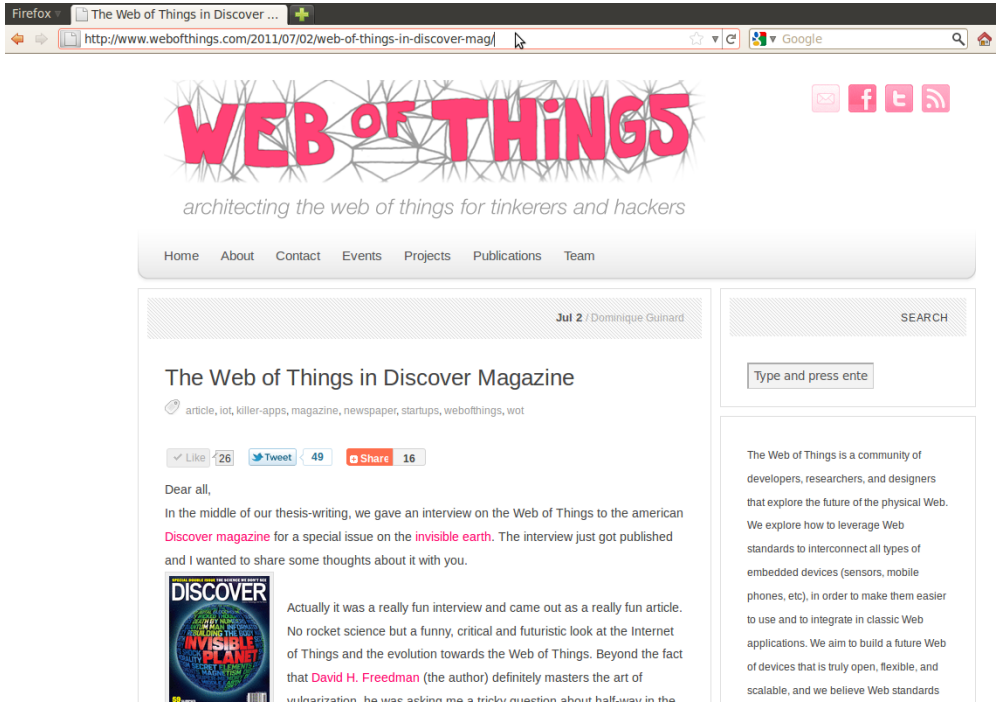
- WoT Application Architecture & evaluation in WSN + RFID
  - The Web can be leveraged and adapted as a smart thing application architecture
  - Eases the development & brings it closer to non-specialists
  - Unveils integration possibilities:
    - Browser, search engines, social networks, Web languages, mashups, etc.



Limitations & Outlook

- Not the best approach for every use-case:
  - Real-time, high QoS requirements, battery-life
- Pushing Internet and Web standards forward:
  - Lower foot-print (6LoWPAN)
  - Web push (HTML5 WebSockets, etc.)
  - Metadata for smart things
- Real-world evaluations:
  - Larger deployments, industrial trials (IPSO alliance)
  - Comparisons with other alternatives
  - Evaluating the mashups with more end-users

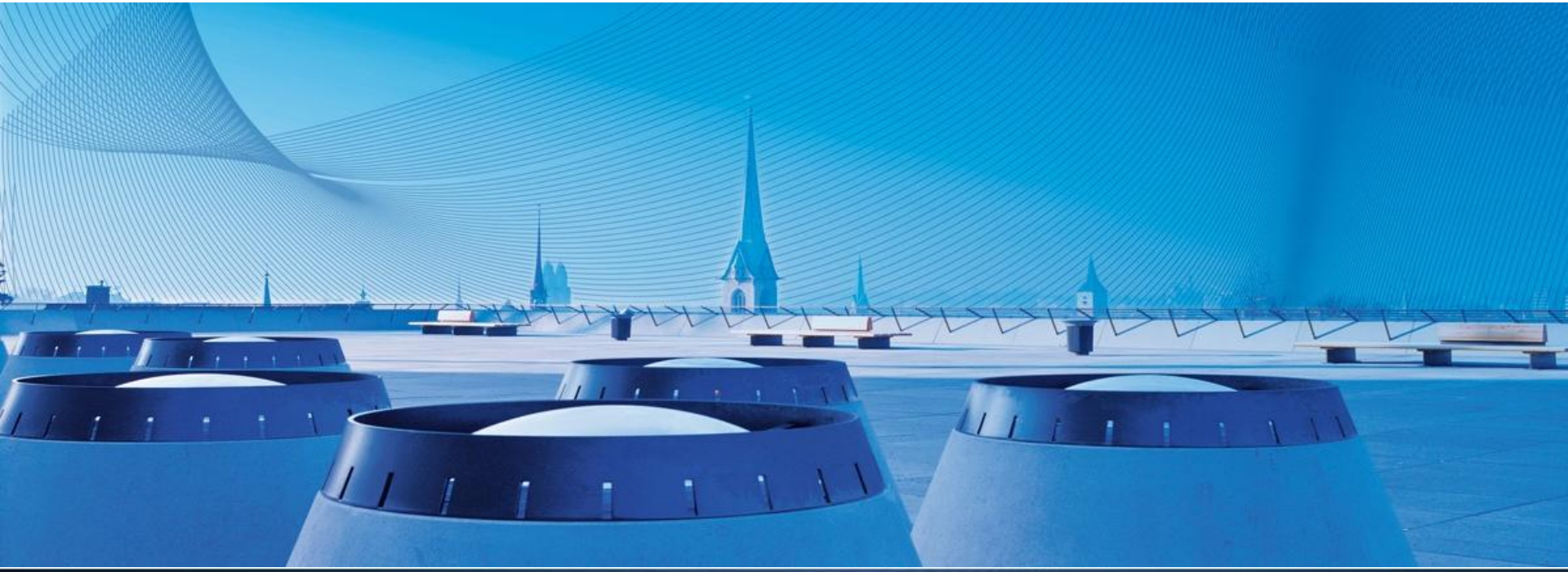
Thanks a lot for your attention



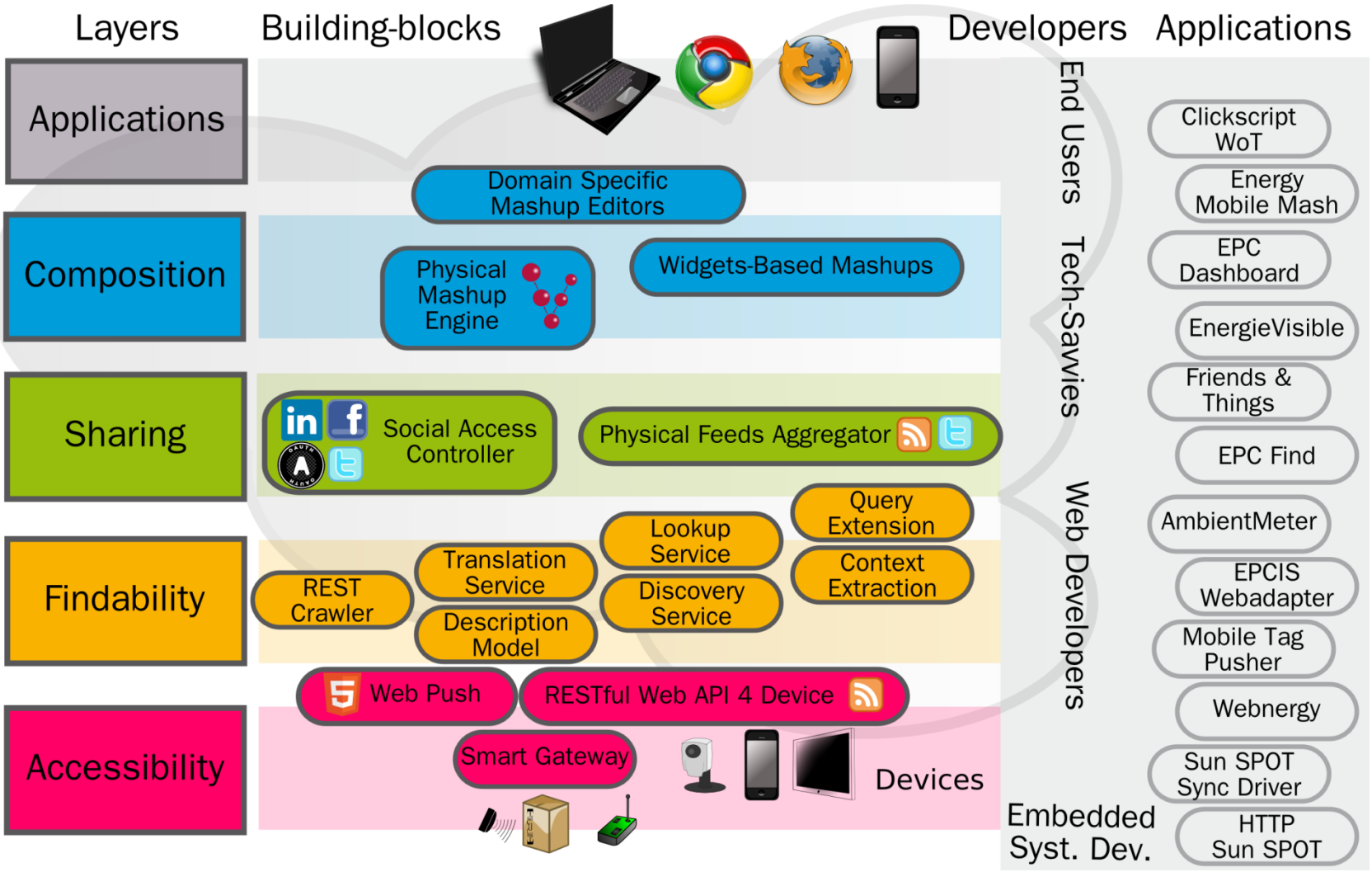
- Dominique Guinard
  - Contact details: www.guinard.org
  - Blog: www.webofthings.com
  - Software: www.webofthings.com/projects
  - Publications www.webofthings.com/publications

Backup

Wait! There is a little more…



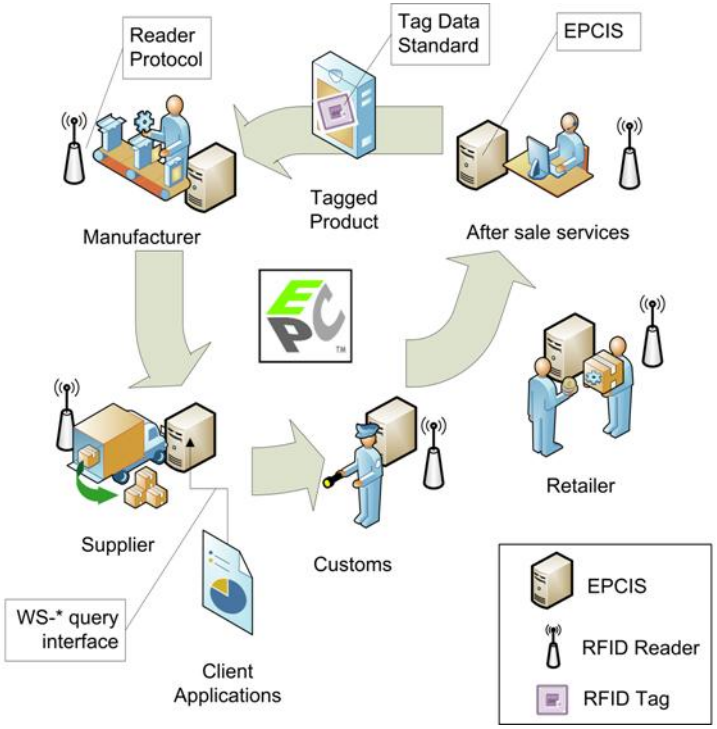
Complete Web of Things Application Architecture



Case-Studies

- Wireless Sensor Networks:
  - General Purpose Sensing Platform (Sun SPOTs)
  - Smart Metering Platform (Ploggs)

- Tagged Objects:
  - RFID global network:
    - EPC Network



Accessibility

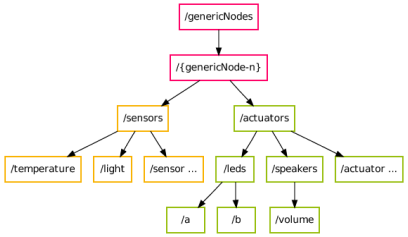
Web-Enabling Smart Things in 4 Steps

Resource Design

Representation Design

Interface Design

Implementation Strategy



Web of Things - Resource Temperature

Home
Parent
Refresh
Atom Feed
Get current temperature: 24.0 C

GET, POST, PUT, DELETE, OPTIONS

Content Negotiation, Status Codes

- Based on REST [Fiel2000] and Resource Oriented Architecture [Rich2007].
- Applied it to smart things in [Gui2010].

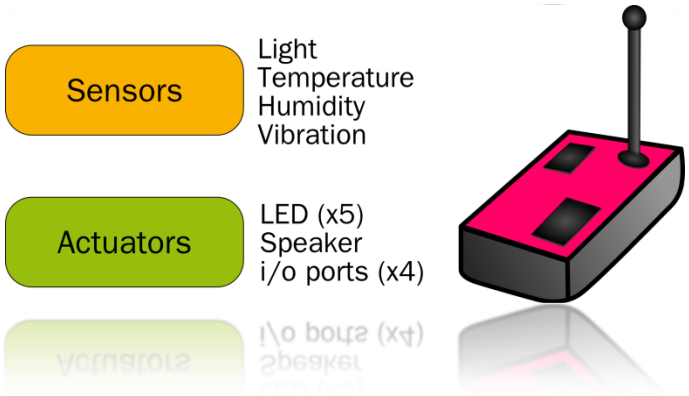
[Fiel2000] Fielding, R. (2000). *Architectural styles and the design of network-based software architectures.* PhD Thesis

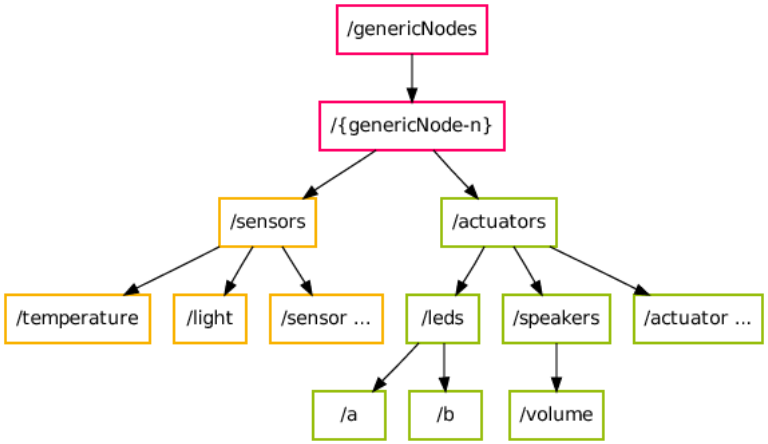
[Gui2010] Guinard, D., Trifa, V., Wilde, E. *A Resource Oriented Architecture for the Web of Things*. IoT 2010

[Rich2007] Richardson, L., & Ruby, S. *RESTful web services*, O□ ReillyMedia.

Accessibility

Resource Design → Representation Design → Interface Design → Implementation Strategy





- Identify Resources:
  - Any component of an application that needs to be used and addressed.
- Link resources together

Accessibility

Resource Design → Representation Design → Interface Design → Implementation Strategy

Web of Things - Resource Temperature

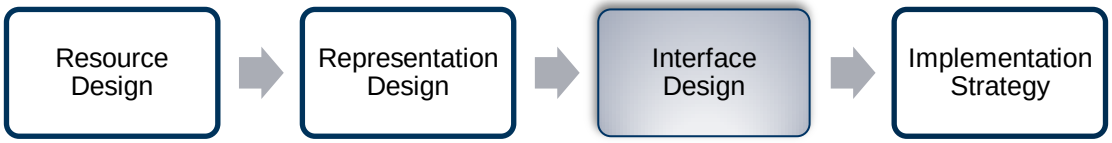
Home
Parent
Refresh
Atom Feed

Get current temperature: 24.0 C

```
{"resource":
  {"methods":["GET"],
  "name":"Temperature",
  "links":["/feed", "/rules"],
  "content":
  [{"description":"Current
Temperature",
  "name":"Current Ambient Temperature",
  "value":"24.0",
  "unit": "celsius"}]}
}
```

- Smart things should offer different representations:
  - HTML for browsability
  - JSON for mashups
  - XML for interoperability

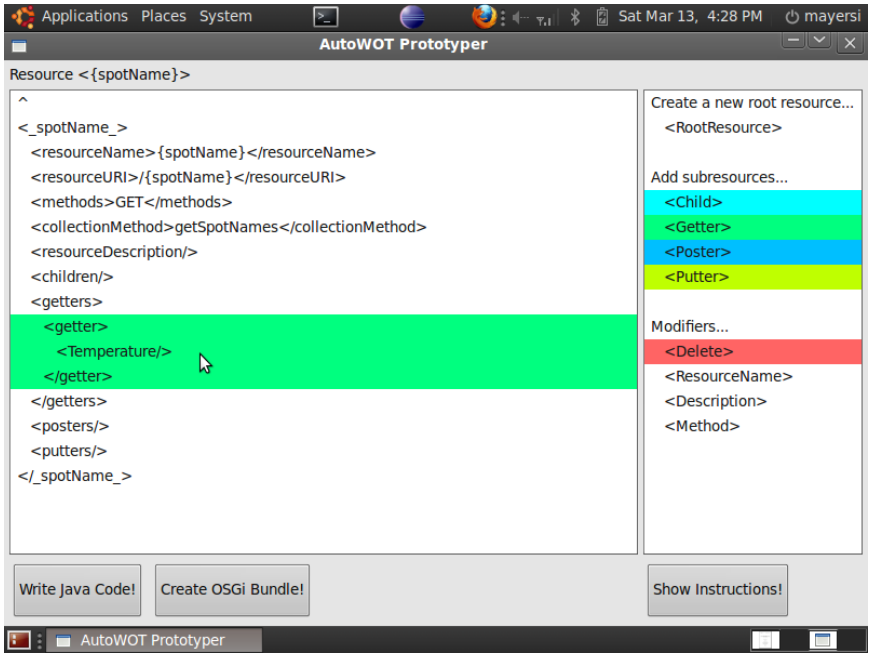
Accessibility



- Leverage content negotiation:
  - Accept: application/json
- Use the HTTP Verbs extensively:
  - GET, PUT, POST, OPTIONS, DELETE
  - GET */genericNodes/2/sensors/temperature*
  - PUT */genericNodes/2/actuators /led/1*
- Map status codes:
  - 200 OK, 201 Created, 400 Bad Request, etc.
- The presented design process can be automated

Automating the Design Process

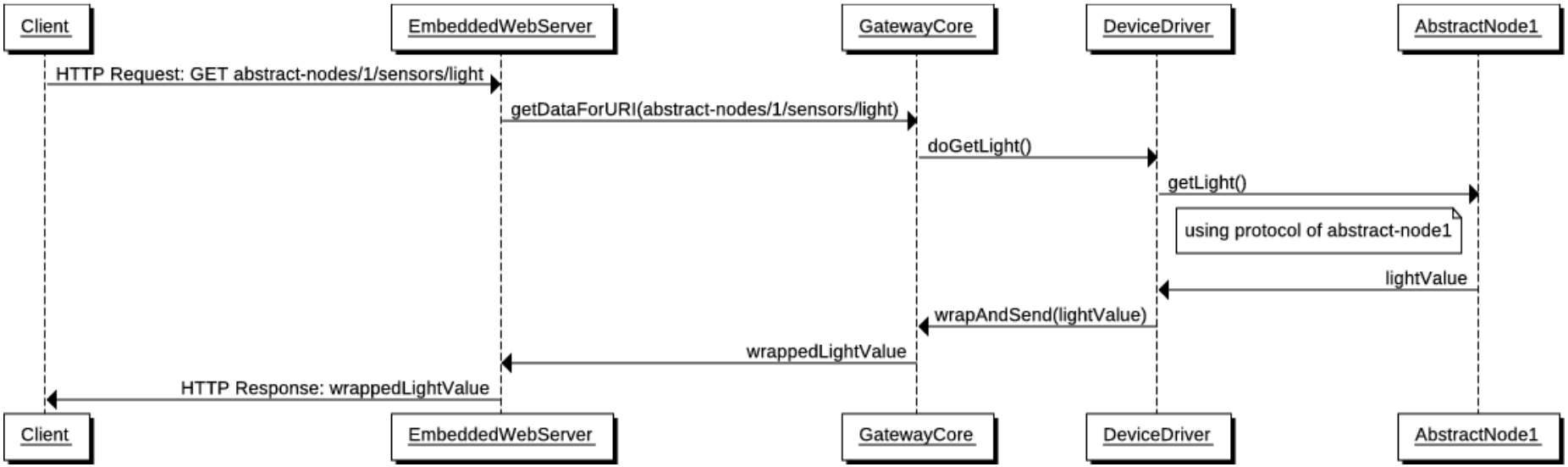
- The design process can be semi-automated through an editor
- Auto-WoT [May2010a] project:
  - Generates the RESTful Web API
  - Wraps it in an OSGi module, loadable as a device driver in a Smart Gateway



[May2010a] Simon Mayer, Dominique Guinard, Vlad Trifa *Facilitating the Integration and Interaction of Real-World Services for the Web of Things*. UrbanIOT 2010

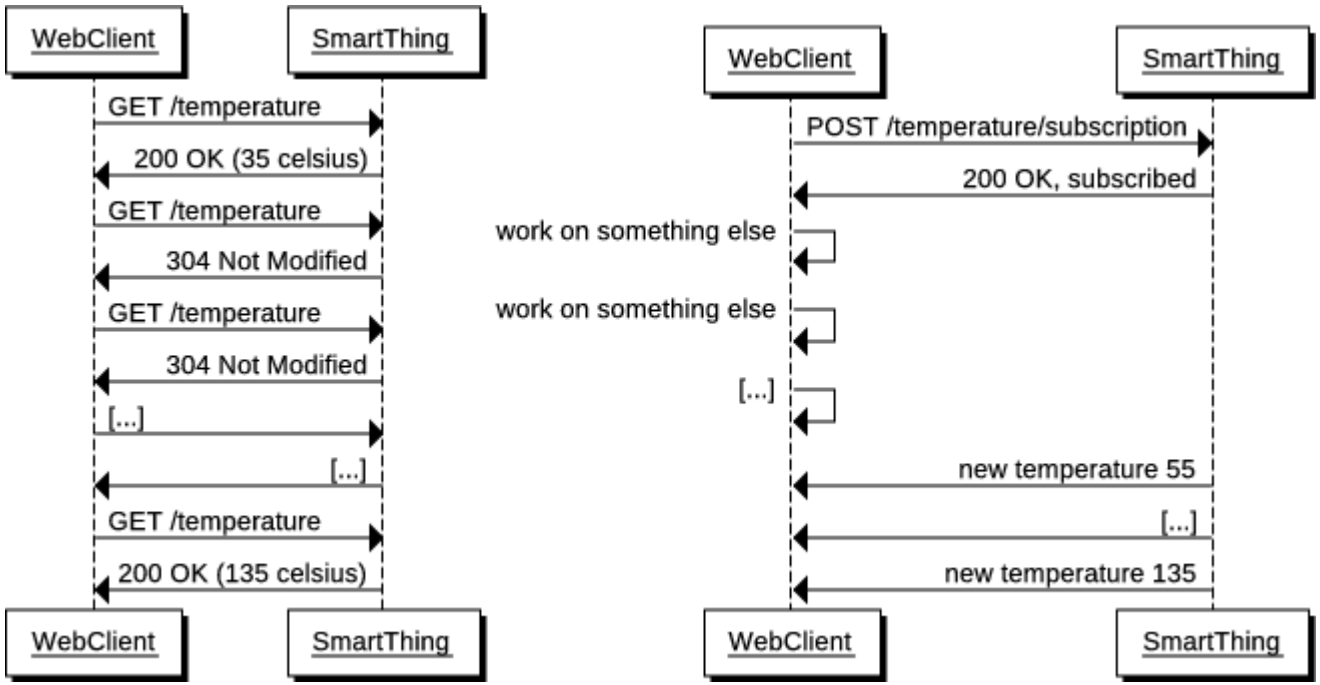
Accessibility

Smart Gateway Mediated Interaction

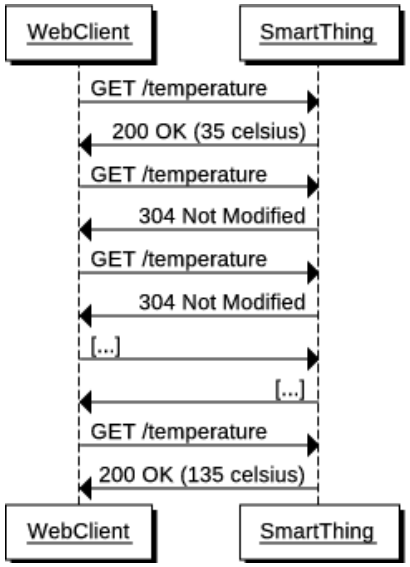


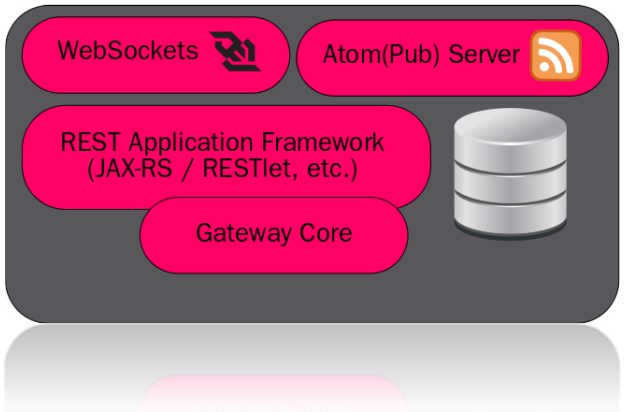
Accessibility

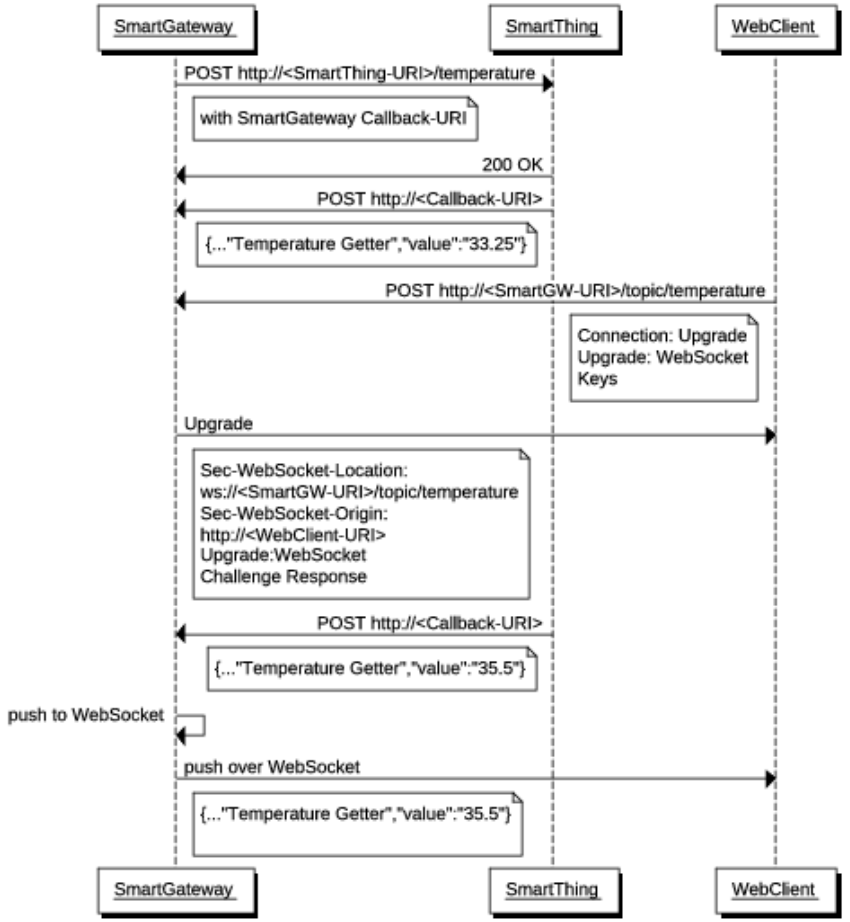
Pushing Data From Smart Things



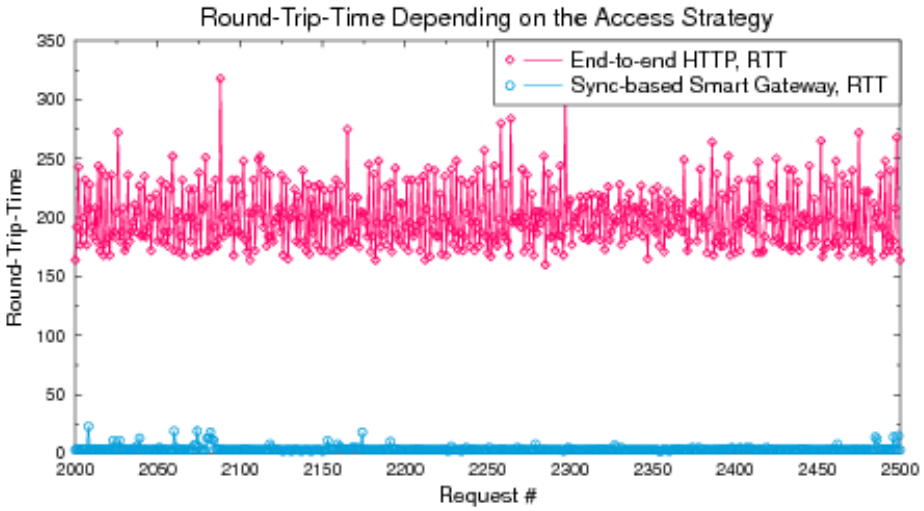
Pushing From Web Sockets: t-Pusher Service

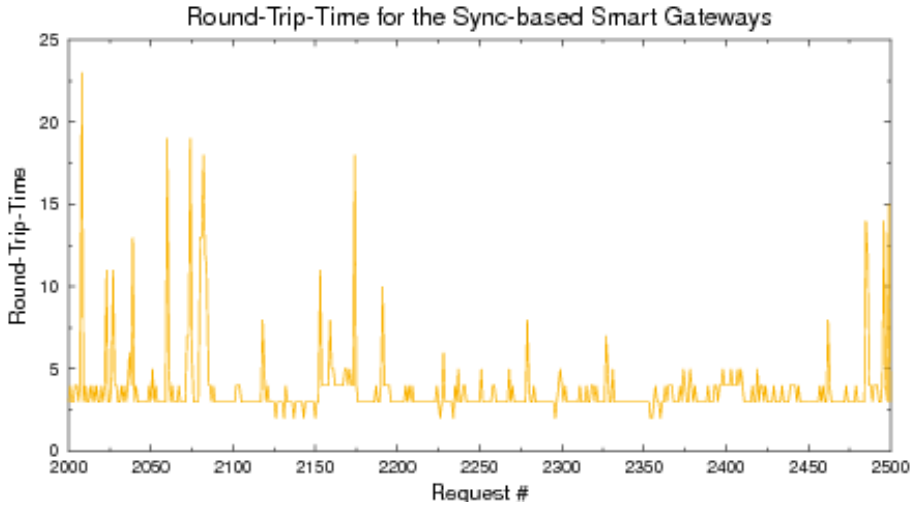






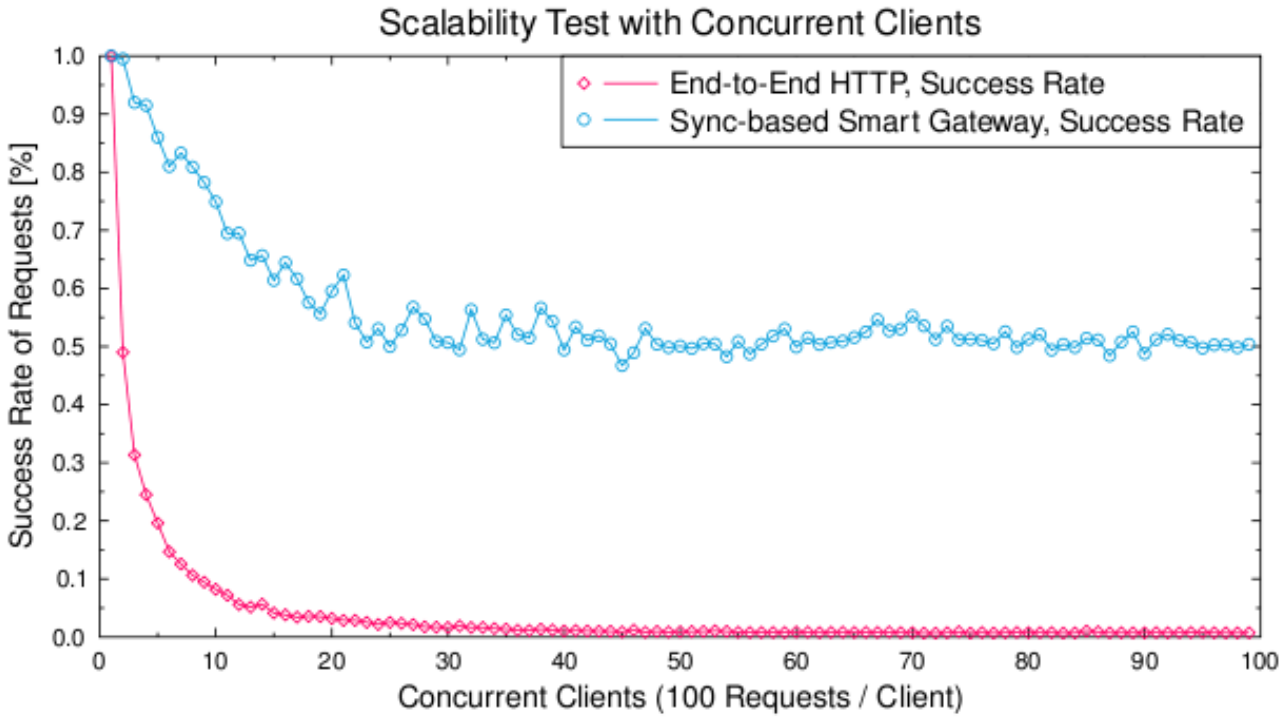
Performance Evaluation





- 7000 (sequential) requests:
  1. End-to-end HTTP
     - Avg.: 205 ms, SD: 127.8 ms
     - Max.: 8.5 sec (single request)
     - Min.: 96 ms
  2. Syn-based Smart Gateway:
     - Avg.: 4.2 ms, SD: 3.7 ms
     - Max.: 111 sec
     - Min.: 2 ms

Scalability / Concurrency Evaluation

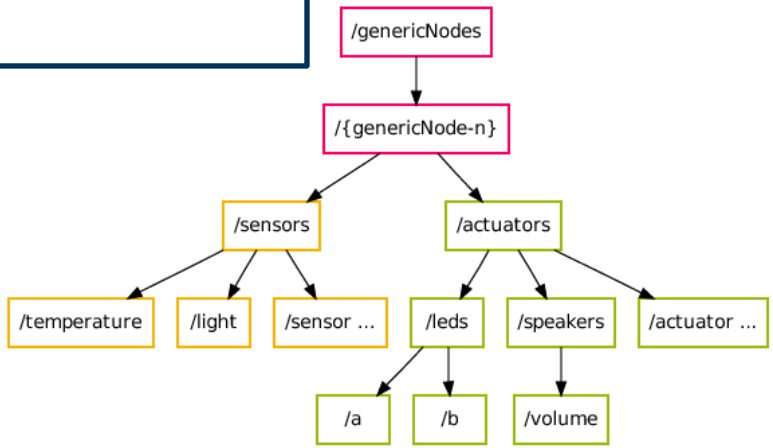


- Smart Gateway sync-based approach
  - Scales better:
    - Strongly depends on the Web server implementation
  - Provides more aged data.

Discovery by Crawling

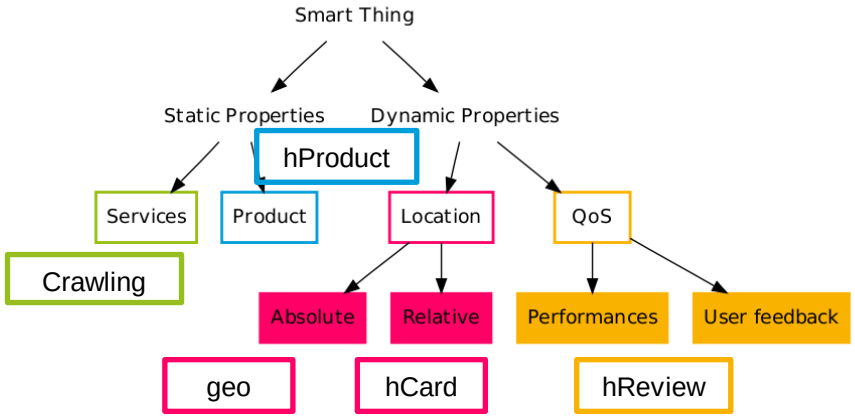
```
crawl(Link currentLink) {
  new Resource() r;
  r.setUri = currentLink.getURI();
  r.setShortDescription = currentLink.text();
  r.setLongDescription = currentLink.invokeVerb(GET).
     extractDescriptionFromResults();
  r.setOperations = currentLink.invokeVerb(OPTIONS).
     getVerbs();
  foreach (Format formats: currentFormat) {
    r.setAcceptedFormats =
     currentLink.invokeVerb(GET).
     setAcceptHeader(currentFormat);
  }
  if (currentLink.hasNext())
     crawl(currentLink.getNext());
}
```

- Thanks to the Accessibility Layer (REST), smart things can be crawled
- Shortcomings:
  - Rough descriptions (UIs)
  - Does not enable automated mashup integration



Findability

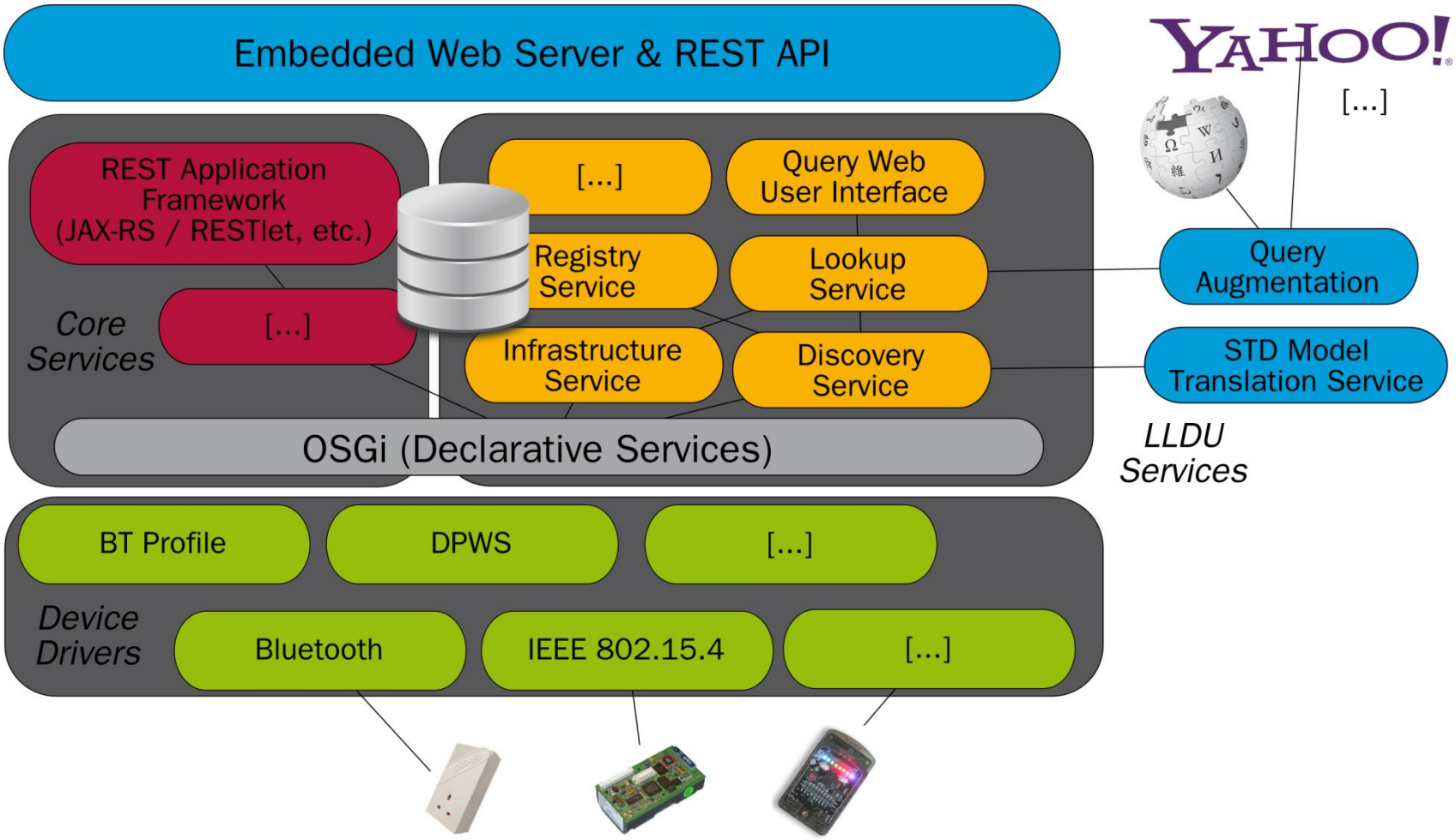
Metadata Description: Smart Things Description Model



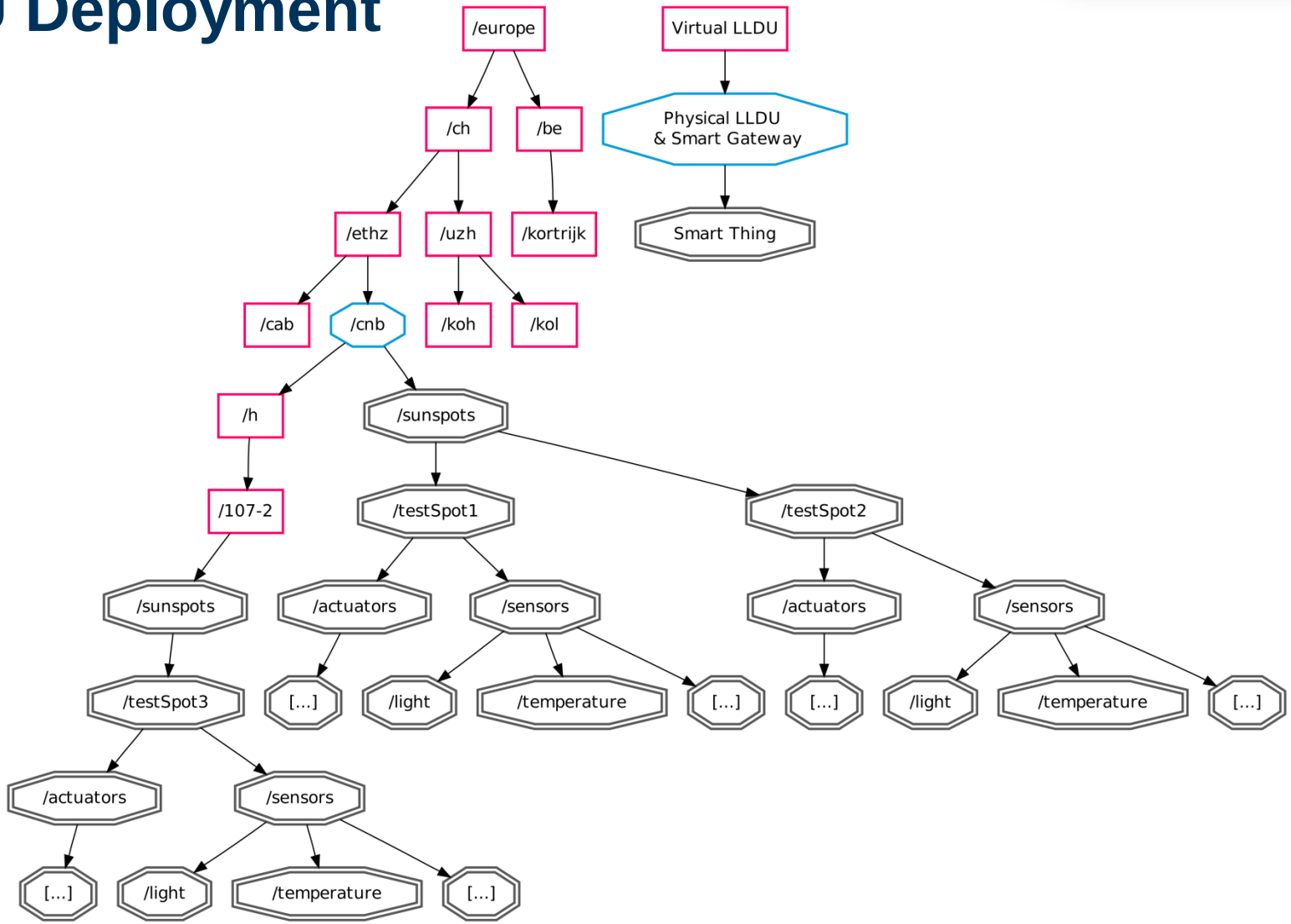
- Description model for:
  - Findablility through existing search engines
  - Automatic integration into mashups
- Crawling algorithm gathers the minimal information:
  - Read URLs
  - GET & OPTIONS on resource
  - Content-negociation
- Compound of microformats: hProduct, hCard, geo, hReview

Findability

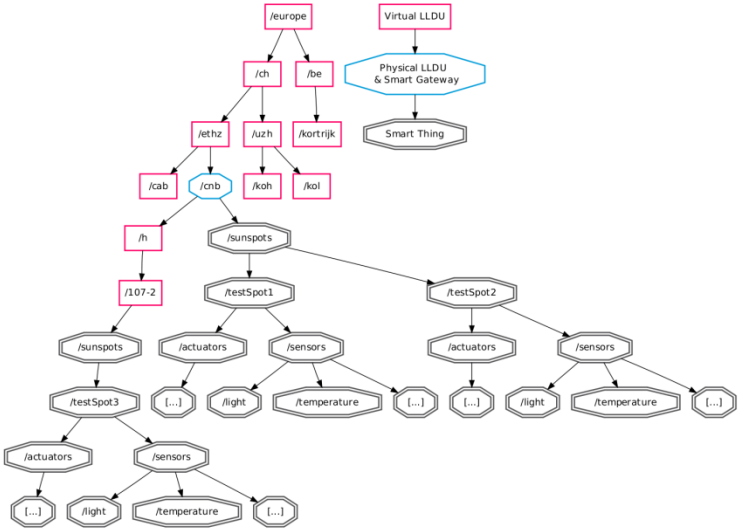
Local Lookup and Discovery Units (LLDUs)

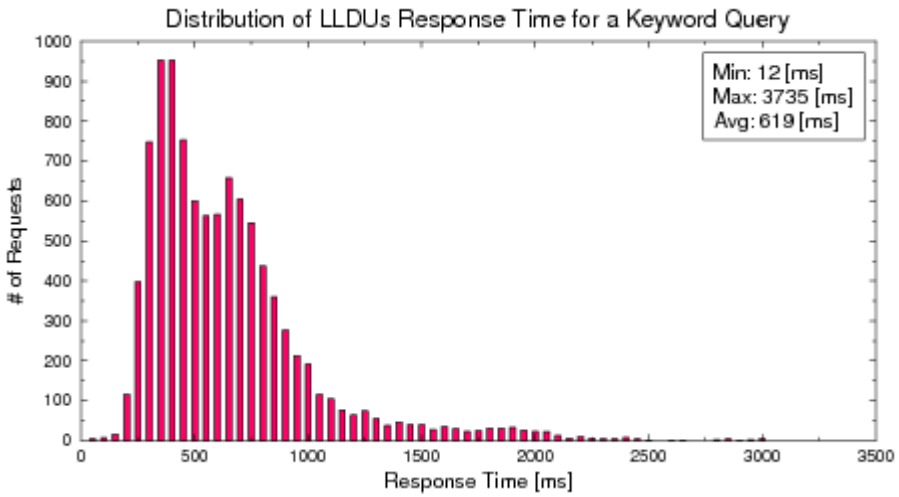


LLDU Deployment



Infrastructure Performance Evaluation

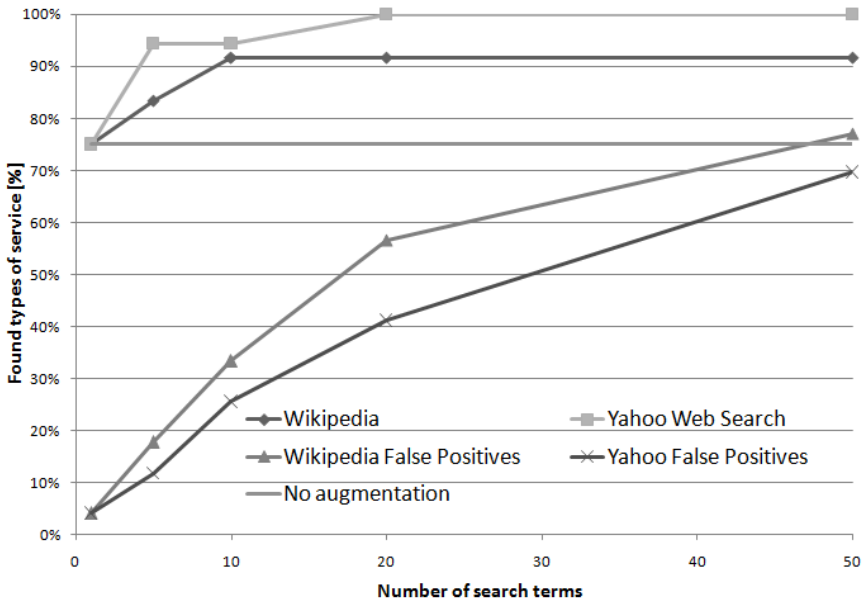




- LLDU tree:
  - Depth of 6
  - 11 virtual LLDUs
  - 1 Physical LLDU (LLDU + Smart Gateway)
    - 2 Sun SPOTs
    - 61 virtual services
- LLDU on a PC (2.4 GHz, 2 GB of RAM):
  - 10000 keywords queries (“light”)
  - Avg: 619 ms
  - Min: 12 ms
  - Max: 3735

Findability

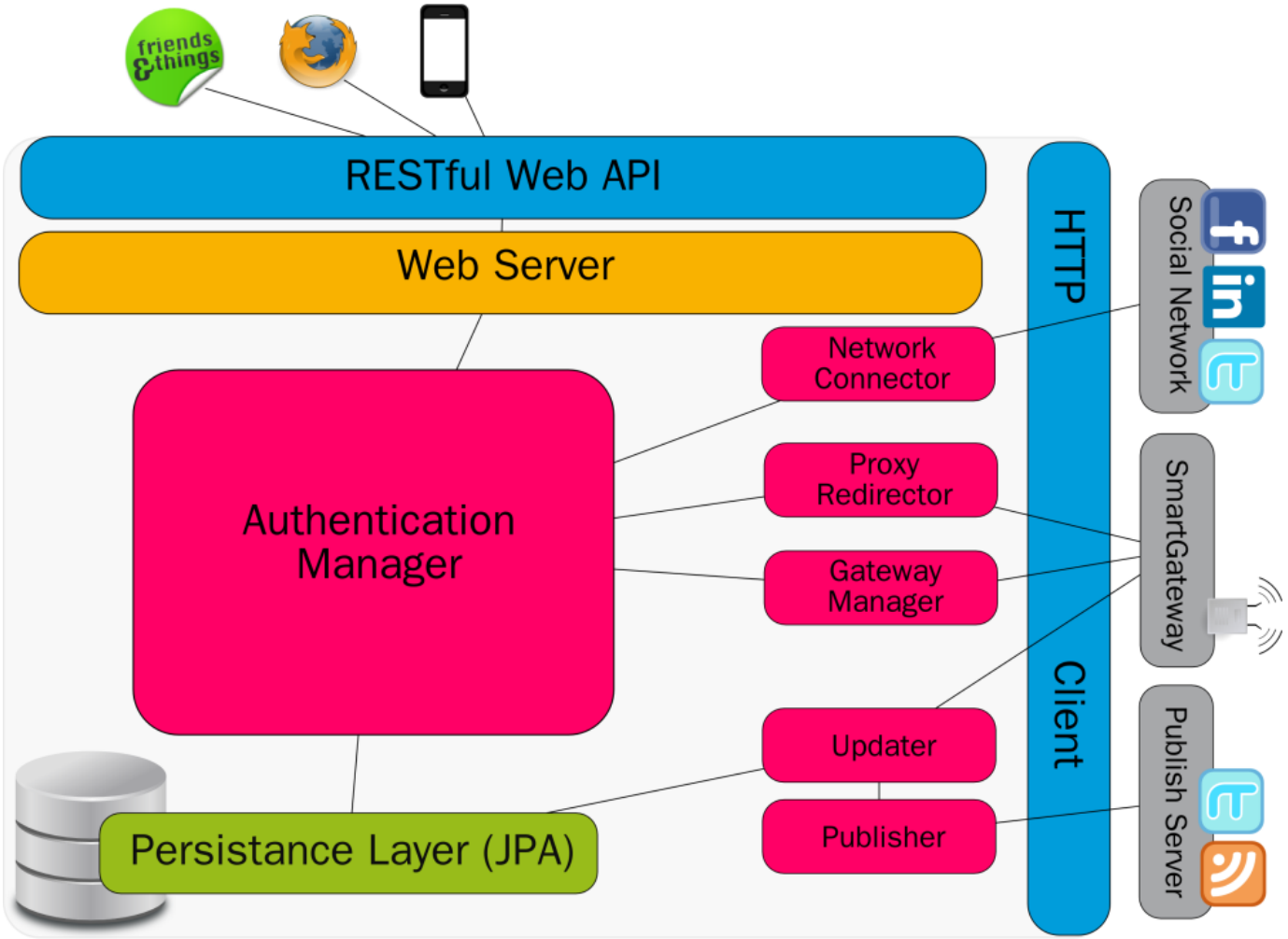
Query Extension Evaluation



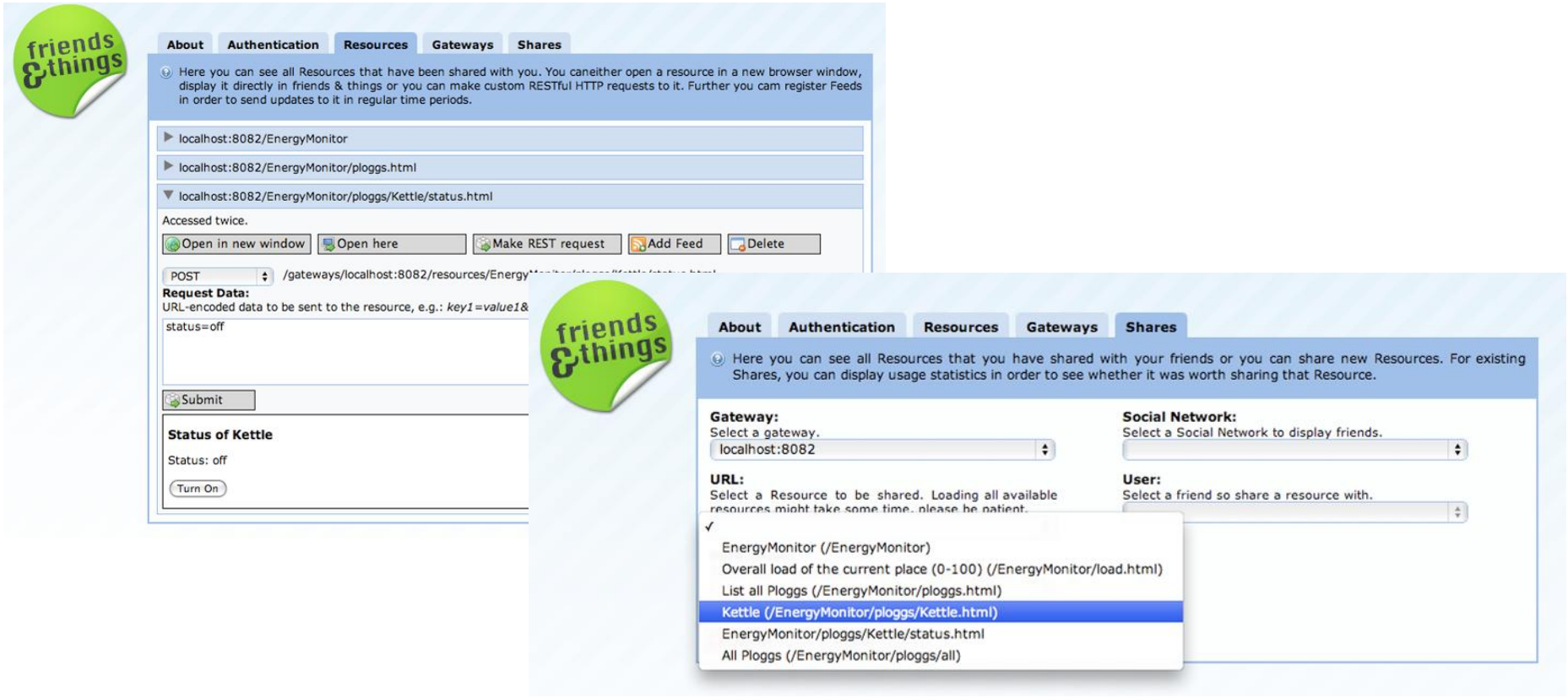
- 17 neutral developers:
  - Each describes one device and at least two services (based on the STD model)
  - RFID readers, robots, smart meters, etc.
- Search keywords provided by 7 IT people:
  - No augmentation: 70%
  - Wikipedia: ~90%
  - Yahoo Web Search: 95-100%
  - Optimum: Yahoo with 5-10 added keywords (95%).

Sharing

Social Access Controller (SAC) Architecture

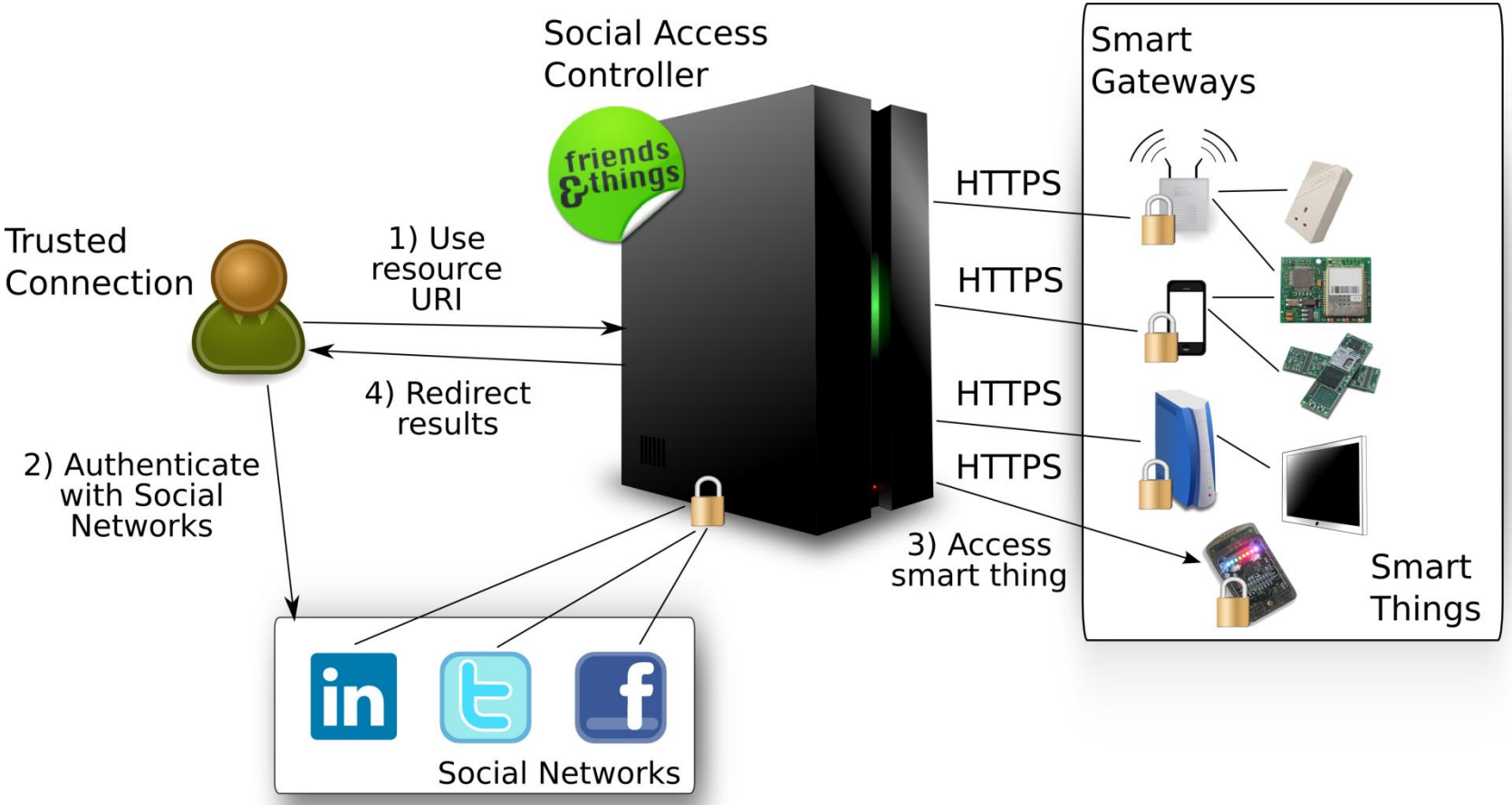


Friends and Things: User Interface



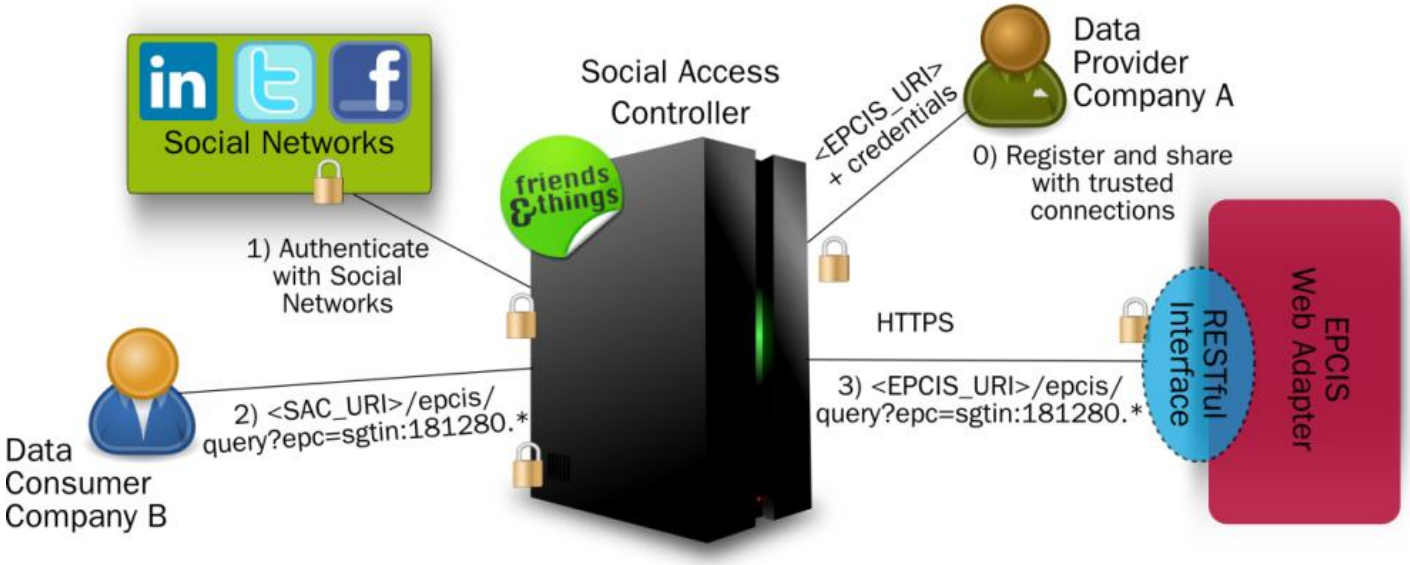
Sharing

Accessing Shared Smart Things: WSN



http://vswot.inf.ethz.ch:8091/gateways/vswot.inf.ethz.ch:8081/resources/sunspots/spot1/sensors/temperature

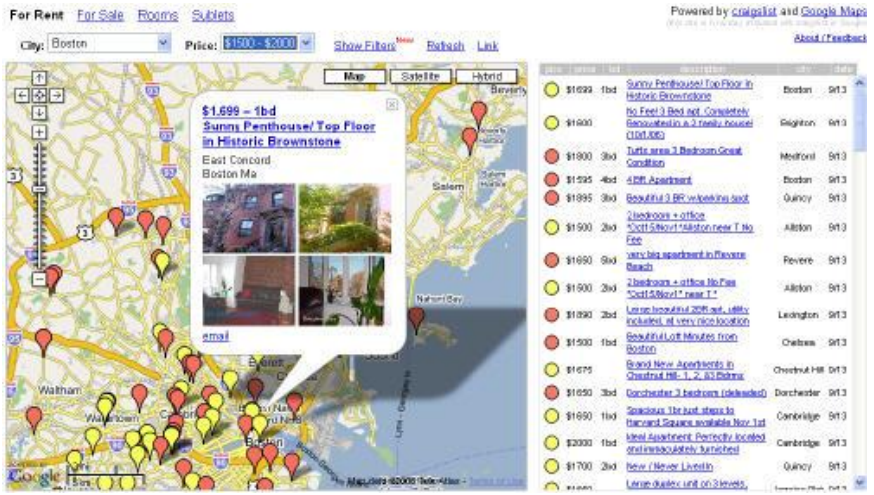
Accessing Shared Smart Things: RFID



http://vswot.inf.ethz.ch:8091/gateways/vswot.inf.ethz.ch:8080/resources/epcis-webadapter/rest/1/eventquery/result?epc=urn:epc:id:sgtin:181280.*

From Web 2.0 Mashups to…

- *Web 2.0 Mashups:*
  - *"Web applications generated by combining […] disparate Web sources […] to create useful new applications or services"* [Yu2008]
- Composite applications with:
  - Lightweightness and simplicity
  - Acessibility to larger public
  - Prototypical or opportunistic nature

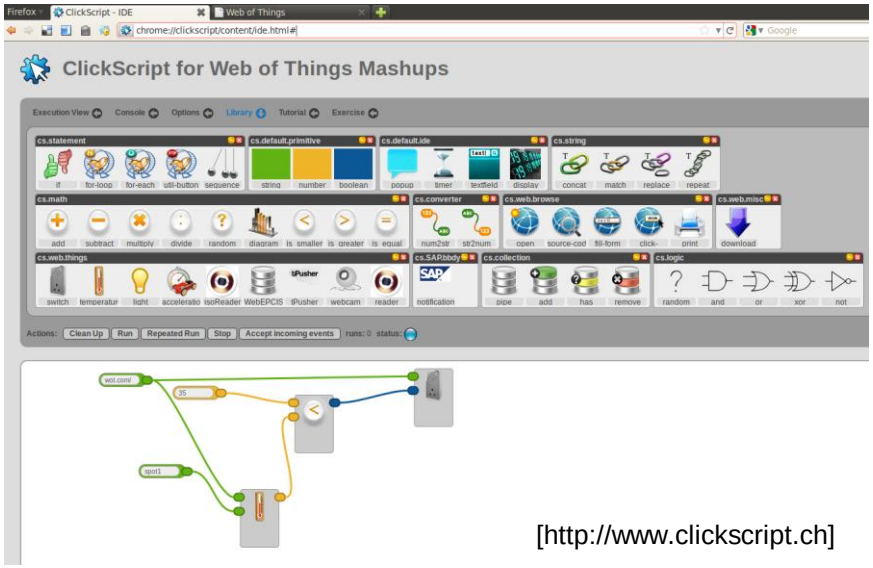


[http://www.housingmaps.com]

[Yu2008] Yu, J., Benatallah, B., Casati, F., & Daniel, F. *Understanding Mashup Development*. IEEE Internet Computing

Manual Mashups with Energie Visible

Adapting a Mashup Editor

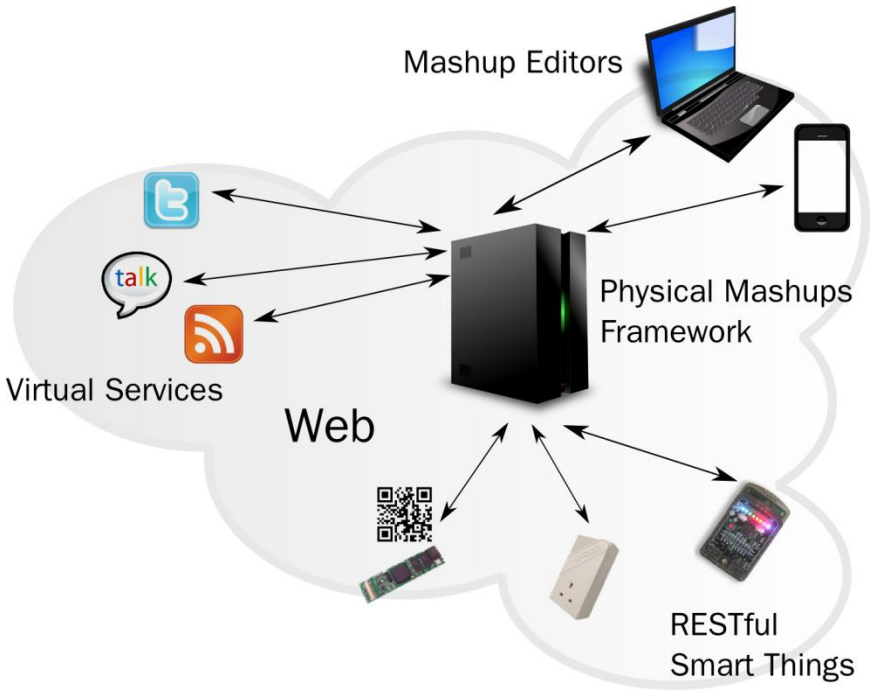


[http://www.clickscript.ch]

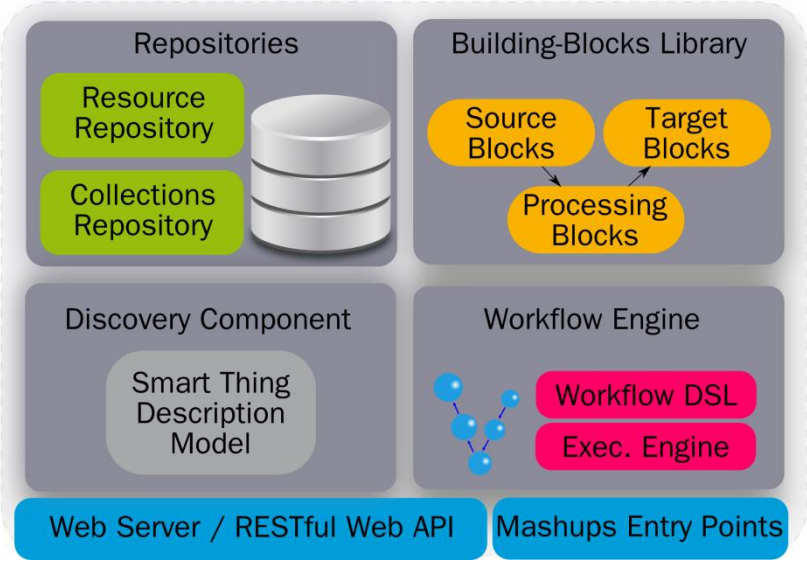
```
$.ajax({
  url: "http://" + ip + "/sunspots/" +
    name + "/sensors/temperature",
  type: "GET", dataType: "json",
  success: function(result){
   var temperature =
    result.resource.getters[0].value
   state.outputs.item(0).setValue(temp)
   component.finishAsync();
  } […]});
```

[Naef2009] Naef, L. *ClickScript a visual programming language in the browser*. Master Thesis, ETH Zurich

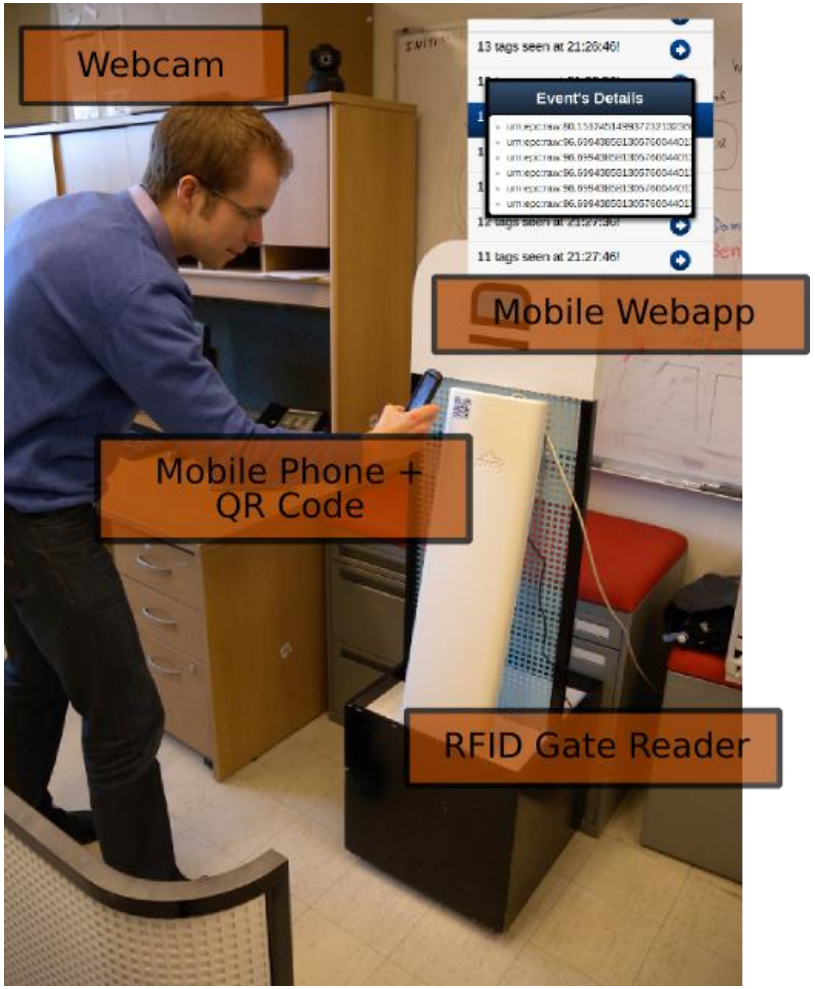
Building Mashup Editors: Physical Mashups Framework

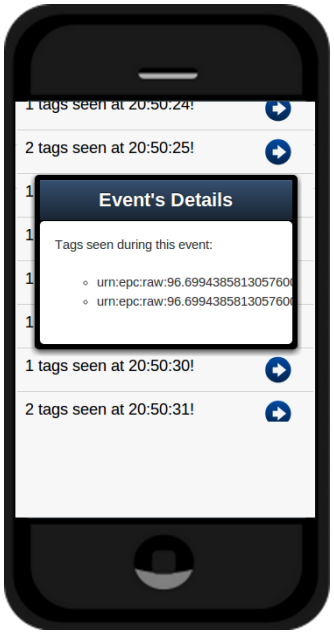


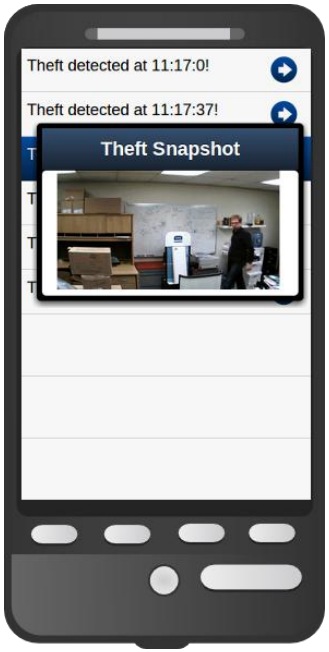
- Requirements:
  - Support for event-based mashups
  - Support for dynamic building-blocks
  - Support for non-desktop platforms
  - Support for application specific editors



Physical Mashup Lab Deployment & Mobile Apps





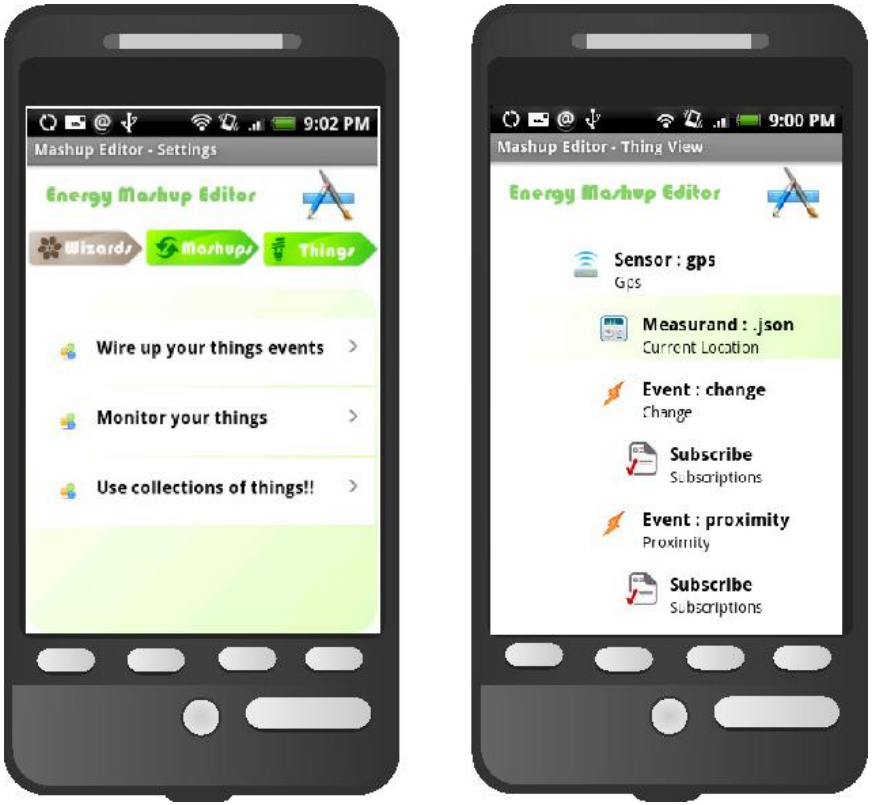


Mobile Energy Mashup Editor

- Android-based mashup editor for mashable homes
- Uses the Physical Mashup Framework RESTful Web API
- Findability Layer for automatic building-blocks generation

[Guinard2010c] Guinard, D. *Mashing up your web-enabled home*. ICWE 2010

Mobile Energy Mashup Editor cont'd



- Lets end-users create simple rules to optimize their energy consumption:
  - Turn the heating on only when I am driving home and temp < 18 deg.
  - Turn off the lights when sun light is strong enough.
- Android-based mashup editor.
- Uses the Physical Mashup Framework RESTful Web API.
- Uses the Findability Layer for automatic building-blocks generation.

Mobile Energy Mashup Editor cont'd

- JSON reflection to generate adapted UIs.

REST vs WS-*: Guidelines

Requirement	REST	WS-*	Justification
Mobile & Embedded	+	-	Lightweight, IP/HTTP support
Ease of use	++	-	Easy to learn
Foster third-party adoption	++	-	Easy to prototype
Scalability	++	+	Web mechanisms
Web integration	+++	+	Web is RESTful
Business	+	++	QoS & security
Service contracts	+	++	WSDL
Adv. security	-	+++	WS-Security