REST or WS-*? A Developer's Perspective

Dominique Guinard, Iulia Ion, and Simon Mayer
MobiQuitous 2011, Copenhagen, Denmark

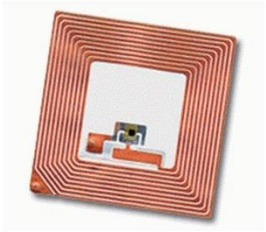
Simon Mayer [simon.mayer@inf.ethz.ch]
Distributed Systems Group, ETH Zurich

Motivation

- Internet of Things
  - Integrating functionality from multiple smart things is hard: It requires expert knowledge from many domains

- Therefore: Smart Things as Service Platforms
  - Provided services should be easy to compose
  - Create a loosely coupled ecosystem of smart things
  - Main approaches: WS-* and REST

Towards a Web of Things









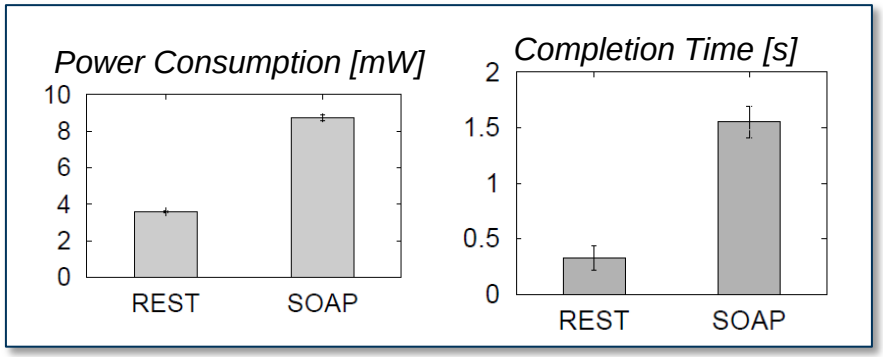






REST or WS-*? Previous Comparisons

- REST and SOAP on devices with limited resources[1]



- RESTful vs. "Big" Web Services[2]
  - REST for tactical, ad hoc integration over the Web ("*Mashup*")
  - WS-* in professional enterprise application integration scenarios with a longer lifespan and advanced QoS requirements

[1] D. Yazar and A. Dunkels: *Efficient Application Integration in IP-based Sensor Networks*

[2] C. Pautasso, O. Zimmermann, and F. Leymann: *RESTful Web Services vs. "Big" Web Services. Making the right architectural decision*

REST or WS-*? Beyond Performance

- User Acceptance: Performance is not enough, rather consider perceived ease of use as the key to adoption of an IT system

- Increasing reliance on external developers to build innovative services (App Store, Android Marketplace)

→ Easy to learn and easy to use API is key to foster a broad community of developers

F. D. Davis: *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*

D. Gefen and M. Keil: *The Impact of Developer Responseiveness on Perceptions of Usefulness and Ease of Use*

REST or WS-*? A Developers' Perspective

- Study to evaluate the developers' experience when learning and implementing Web Service applications
  - Perceived advantages & disadvantages of REST and WS-*
  - Ease and speed of learning
  - Perceived suitability of REST and WS-* for different scenarios

- Participants
  - Computer science students (n = 69)
  - Third or fourth year of Bachelor studies
  - Previous knowledge: None of WS-* (89%), None of REST (62%)

Study Setup: Data Sources

- Data Sources (quantitative and qualitative feedback)
  1. Implementation of Mobile Phone Apps — Teams (n = 25)
  2. Structured Questionnaire — Individual (n = 69)
  3. Feedback Form — Anonymous (n = 37)

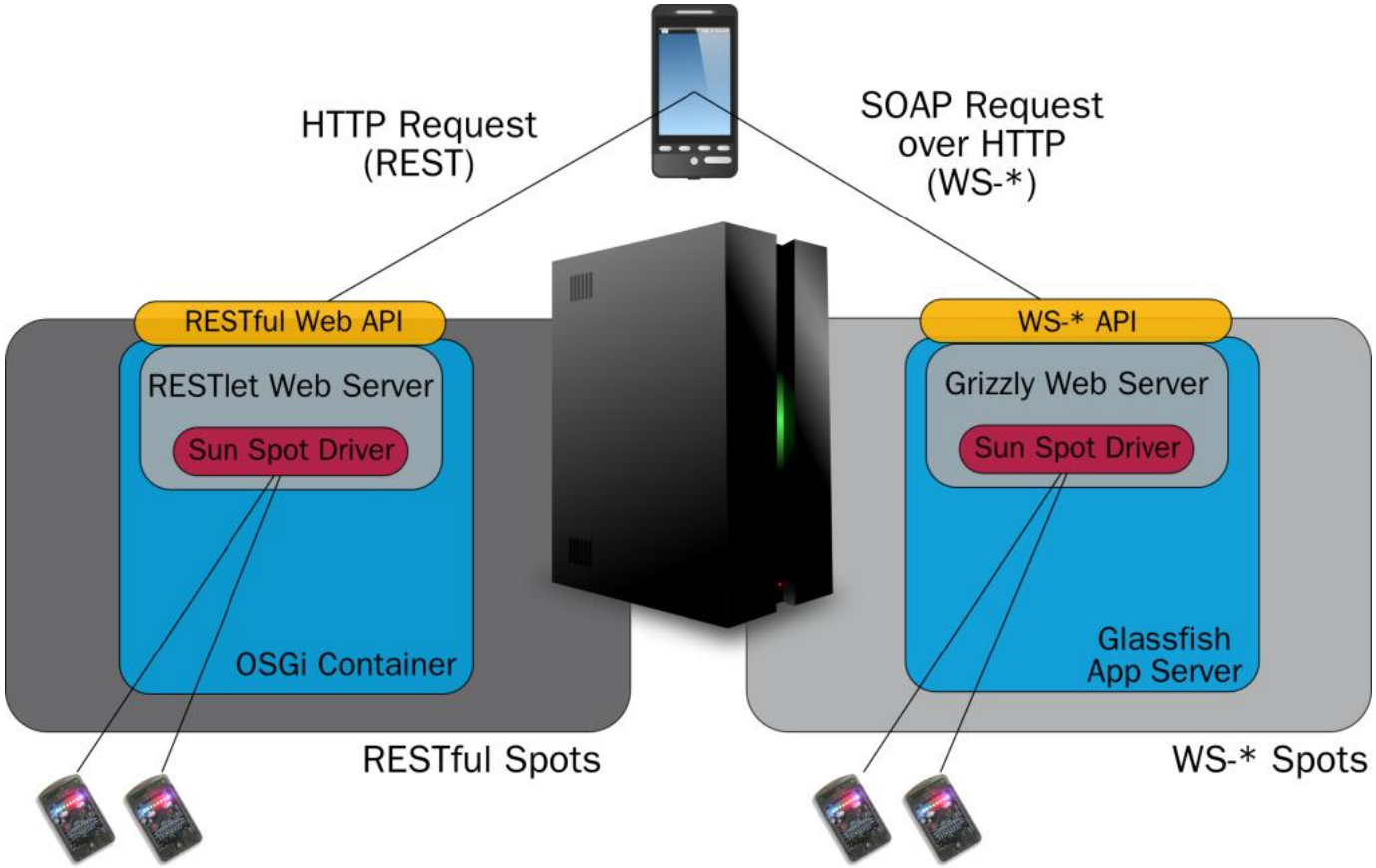
1. Implementation Task: Mobile Phone Applications (n = 25)

- Access temperature measurements on wireless sensor nodes
  a. RESTful API
  b. WS-* (WSDL + SOAP) API

- Standard libraries: Apache *HTTPClient* for REST, *kSoap2* for WS-*

All course material is available online: *tinyurl.com/vs-material*

Study Setup: Data Sources



All course material is available online: *tinyurl.com/vs-material*

Study Setup: Data Sources

2. Structured Questionnaire (n = 69)
   - Qualitative: Advantages and disadvantages of REST and WS-*
   - Questions related to use case scenarios and suitability of WS-* and REST in different domains
   - Completed after finishing implementation

3. Anonymous Feedback Form (n = 37)
   - Questions related to WS-* / REST ease and speed of learning
   - Questions related to suitability of WS-* / REST in different domains
   - Completed after finishing implementation

Results: Overview

- Perceived advantages of each technology
- Ease and speed of learning
- Perceived suitability for use cases
  - Embedded devices
  - Mobile phone client applications
  - Business applications

Results: Perceived Advantages

Qualitative Results, n = 69

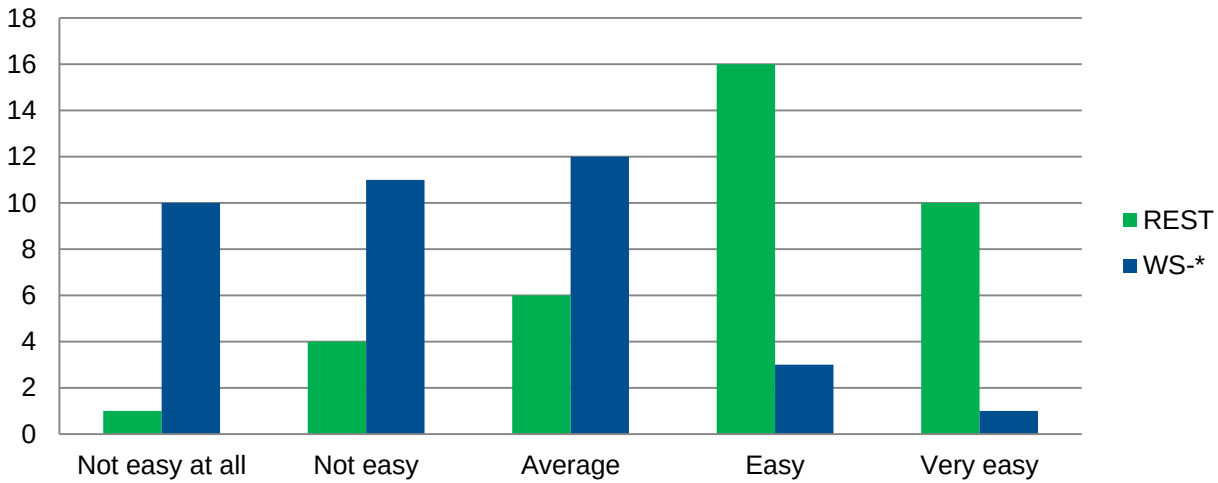
- REST
  - Very easy to understand, learn, and implement (36 participants)
  - More lightweight (27)
  - More scalable (21)
- WS-*
  - WSDL enables service contracts (31)
  - Better security features (19)
  - Better level of abstraction (11)

REST ($N = 69$)	#
Easy to understand, learn, and implement	36
Lightweight	27
Easy to use for clients	25
More scalable	21
No libraries required	17
Accessible in browser and bookmarkable	14
Reuses HTTP functionality (e.g., caching)	10
WS-* ($N = 69$)	#
WSDL allows to publish a WS-* interface	31
Allows for more complex operations	24
Offers better security	19
Provides higher level of abstraction	11
Has more features	10

Results: Ease of Learning

5 point Likert scale [1 = *Not easy at all*, ..., 5 = *Very easy*], n = 37

- "Easy" or "Very easy" to learn
  - REST: 70%
  - WS-*: 11%

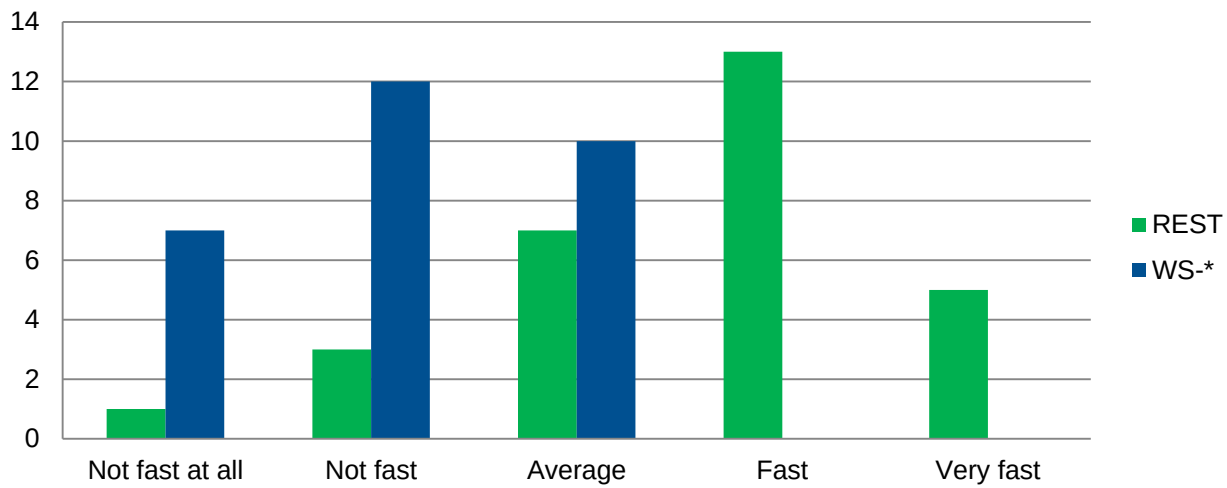


- Statistics (n = 19, accounted for task participation)
  - REST M = 3.85, SD = 1.09
  - WS-* M = 2.50, SD = 1.10
  - REST significantly easier to learn ($p < 0.001$, Wilcoxon signed rank test)

Results: Speed of Learning

5 point Likert scale [1 = *Not fast at all*, ..., 5 = *Very fast*], n = 29

- "Fast" or "Very fast" to learn
  - REST: 65%
  - WS-*: 0%



- Statistics (n = 14, accounted for task participation)
  - REST M = 3.43, SD = 1.09
  - WS-* M = 2.21, SD = 0.80
  - REST significantly faster to learn ($p < 0.009$, Wilcoxon signed rank test)

Results: Ease and Speed of Learning

Qualitative Results, n = 37

"Everybody who is using a browser already knows a little about [REST]"

- REST easier to learn because RESTful Web Services are based on familiar technologies such as HTTP (9)
- REST made it easier to understand what services the sensor nodes offer (25). This is because of the HTML interface (8)
- WSDL and SOAP are more complex to use (8)
- Good that WSDL is "standard" (7)

"REST is easy and WS- is just a complicated mess."*

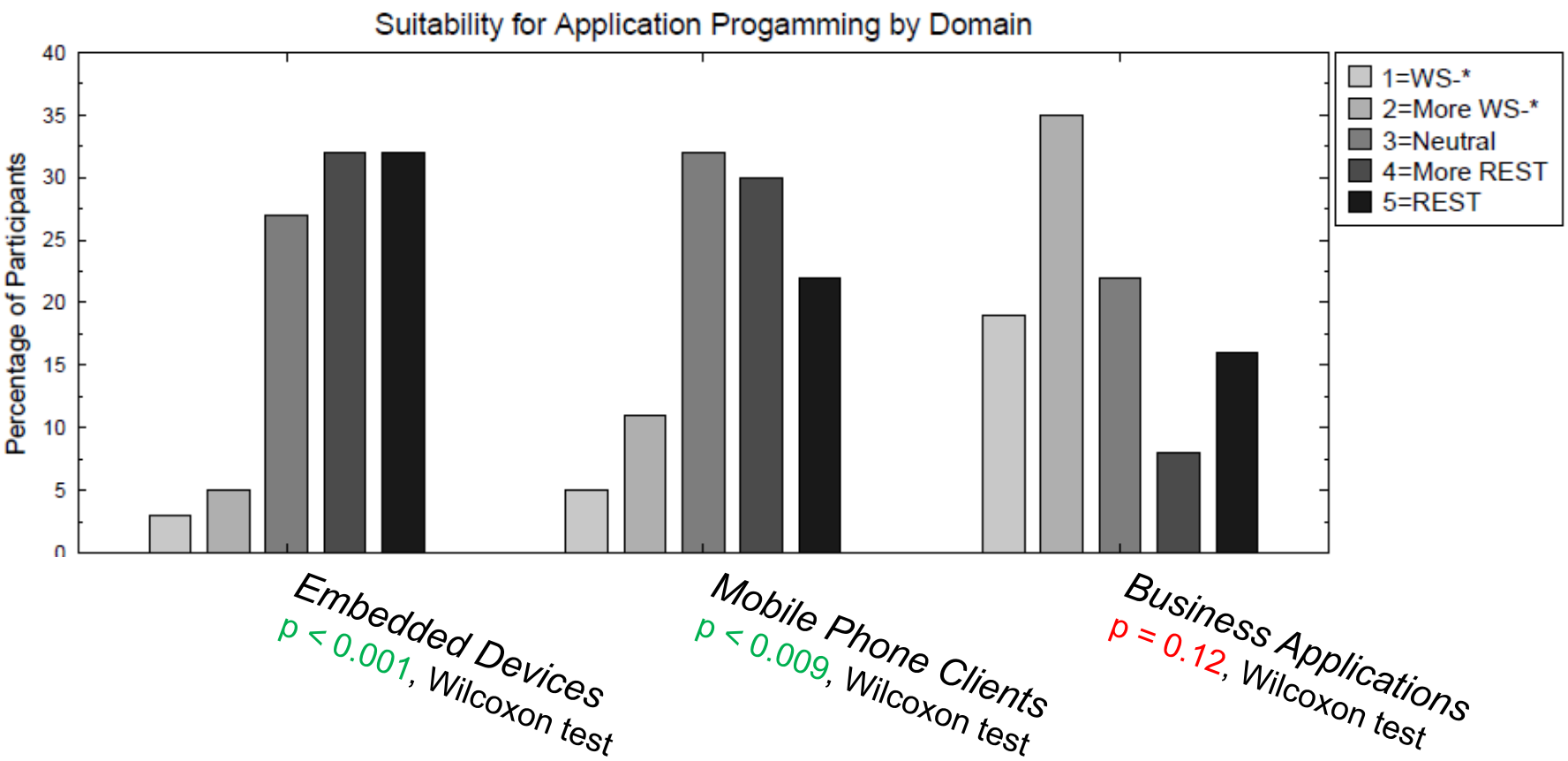
Results: Suitability for Use Cases

Qualitative Results, n = 69

- REST
  - Simple applications requiring only CRUD operations (23)
  - Applications where no higher security level than https is needed (8)
  - Applications that present content directly to the user (6)
  - REST services compose easily (14) → Web Mashups
- WS-*
  - Applications that require extended security features (20)
  - Requirement of strong contracts on the message format (16)

Results: Suitability for Use Cases

5 point Likert scale [1 = *WS-**, ..., 5 = *REST*], n = 37

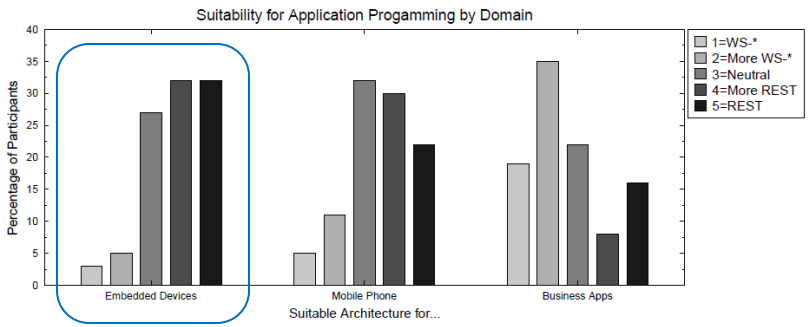


Results: Suitability for Use Cases

5 point Likert scale [1 = *WS-**, ..., 5 = *REST*], n = 37 + qualitative data

- Embedded Devices
  - REST (66%), WS-* (8%)
  - Reasons: REST better in heterogeneous environments, more lightweight *(avg. footprint 17.46 kB for REST, 83.27 kB for WS-* application)*
  - Smart Home Sensor Network (students' private homes)
    - REST (89%), WS-* (7%)
    - REST reasons: Simplicity of deployment and use (24)
    - Surprisingly little security concerns in smart home environments (14)

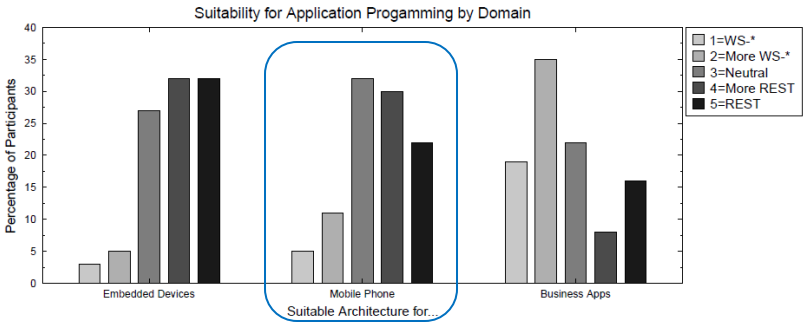
"Information isn't really sensitive"



Results: Suitability for Use Cases

5 point Likert scale [1 = *WS-**, ..., 5 = *REST*], n = 37 + qualitative data

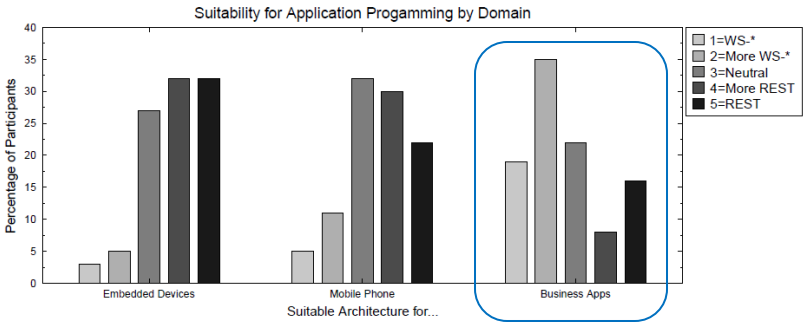
- Mobile Phones
  - REST (53%), WS-* (16%), 32% undecided
  - Reasons: REST causes less traffic (7)
  - Undecided reasons: Mobile phones getting very powerful



Results: Suitability for Use Cases

5 point Likert scale [1 = *WS-**, ..., 5 = *REST*], n = 37 + qualitative data

- Business Applications
  - WS-* (52%), REST (24%)
  - WS-* Reasons: Security needs (21), better service contracts (18)
  - REST Reasons: Simplicity (10), Scalability (10)



Summary

- **REST**: Intuitive, flexible, lightweight
- **WS-***: Advanced security, clearer standardization, service contracts

- Preference for REST for learning ease and speed (significant)

- Preference for REST for embedded applications and applications for mobile phone clients (significant)

- Preference for WS-* for business applications (not significant)

Summary

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

Discussion

- WS-* perceived as being better suited for business applications: Perception bias?
- Surprisingly little security concerns for smart homes!
- REST as relatively fuzzy concept: Many wrong interpretations of the concept!
- Lack of formal service contracts for RESTful Web Services?!

Acknowledgements

Matthias Kovatsch, Web of Things team at ETH

Students of Distributed Systems class 2010

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

FNSNF
FONDS NATIONAL SUISSE
SCHWEIZERISCHER NATIONALFONDS
FONDO NAZIONALE SVIZZERO
SWISS NATIONAL SCIENCE FOUNDATION

WoT 2012

Third International Workshop on the Web of Things

www.webofthings.org/wot

Collocated with Pervasive 2012
June 2012, Newcastle, UK

WEB OF THINGS
architecting the web of things for tinkerers and hackers