<http://www.diva-portal.org>

Postprint

This is the accepted version of a chapter published in *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence*.

Citation for the original published chapter:

Mathew, S S., Atif, Y., Sheng, Q Z., Maamar, Z. (2013)

The Web of Things: Challenges and Enabling Technologies.

In: Nik Bessis, Fatos Xhafa, Dora Varvarigou, Richard Hill, Maozhen Li (ed.), *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence* (pp. 1-23).

Springer Berlin/Heidelberg

Studies in Computational Intelligence

http://dx.doi.org/10.1007/978-3-642-34952-2_1

N.B. When citing this work, cite the original published chapter.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-12830>

The Web of Things – Challenges and Enabling Technologies

Sujith Samuel Mathew^a, Yacine Atif^b, Quan Z. Sheng^a, Zakaria Maamar^c

^a School of Computer Science, University of Adelaide, Adelaide, Australia

^b Faculty of IT, UAE University, Al Ain, UAE

^c College of IT, Zayed University, Dubai, UAE

{sujith, [quanzheng.sheng](mailto:quanzheng.sheng@adelaide.edu.au)}@adelaide.edu.au, yacine.atif@uaeu.ac.ae, zakaria.maamar@zu.ac.ae

Abstract Advances in ubiquitous computing and embedded systems are closing the gap between the physical and virtual worlds. With the Internet becoming the platform of choice for communication, ubiquitous computing faces new opportunities and challenges. Today, things like refrigerators and bathroom scales are becoming Internet enabled and the trend is increasing with Internet penetration predicted to cover a wide range of physical appliances. The *Internet of Things* (IoT) is an active research area, focusing on connecting real-world things over TCP/IP. This trend has recently triggered the research community to ponder the use of the Web (HTTP) as an application platform for integrating things on the Internet. The Web is now evolving from an information portal to an application platform for connecting people and things together. Harnessing physical things into the virtual world using Web standards also enriches the arena of conventional Web services to unleash data and function of physical things to be participants on the information highway. In this chapter, we introduce the *Web of Things* (WoT) as an application platform for the world of heterogeneous things around us. We reveal the state of the art in research and discuss the models and technologies driving the WoT to support future ambient environments where people and things interact seamlessly.

Introduction

Weiser envisioned *Ubiquitous Computing*, where computing becomes invisibly integrated into the world around us and accessed through intelligent interfaces. He observed, “*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*” [1]. This vision is being realized today with the influx of *Smart* home appliances, vehicles, and devices into our world. This becomes possible because of the technology advances in the recent years in the area of communication and embedded systems. The *Internet* is today the underlying communication platform of choice,

which is enabling this fast-paced immersion into the physical world. Internet-enabled mobile phones, televisions, refrigerators, bathroom scales, and pacemakers are paving the way for everyday objects around us such as sidewalks, traffic lights, and other commodities to be identifiable, readable, recognizable, addressable, and even controllable via the Internet (TCP/IP) [2]. These trends are motivating researchers to study, model, design, and implement novel applications, which promote interoperability of heterogeneous physical things on the Internet. The *Internet of Things* (IoT) is a fast growing area of research brimming with new ideas, discoveries, standards and challenges towards encompassing things into the virtual world of the Internet.

While the IoT provides the necessary networking connectivity and communication protocols to reach physical things, a parallel and recent research trend evaluates the Web as an application platform for building ambient environments. The spread of the Internet provides the pervasive environment for the use of things, while research in the *Web of Things* (WoT) is poised to provide the application and service layer for real-world things to interoperate over HTTP [3, 4]. The Web has been an important catalyst in the spread and popularity of the Internet. Its open platform has engaged application providers and users to exchange rich-format information including multimedia information. Existing network infrastructures like Wi-Fi, Ethernet, and 3G leverage the use of Web technologies which facilitate the availability of tools, frameworks and information for developing Web based applications. These factors reduce the learning curve when using standard Web protocols for extending the scope of Web applications to real-world things.

Fig.1 illustrates the high-level approach to realize a Web-enabled thing. Augmenting a thing with Internet connection (i.e., IP address) ensures its accessibility over the Internet and results in an Internet-enabled thing, as shown in Fig.1 (a). When a thing is Internet-enabled and is, also, connected to a Web server, then it becomes Web-enabled, as shown in Fig.1 (b). The Web server may be either embedded or on a separated system. Advances in embedded technology have made it possible for realizing tiny Web servers which are widely available now and fueling the trend towards having Web servers embedded in everyday objects [5].

Fig. 1. Realizing a Web-enabled thing

The wide proliferation of the Web has paved the path for Web services to be an indispensable technology, for interoperable applications. Extending service computing to Web-enabled things makes it possible for real-world things and their operations to be abstracted as Web services. In doing so, real-world things could offer their functionality via SOAP-based Web services (WS-*) or RESTful APIs [6]. These services directly expose real-world things' operations to enterprise applications or even to other things and possibly involving a dynamic composition process at runtime. This research direction of the WoT opens up many opportunities and challenges where real-world things are identified, searched, and composed on the Web. This will eventually lead to the realization of *Ambient Spaces* where people and things seamlessly interact with each other on the Web.

Towards the Web of Things

The idea of integrating and networking real-world things to communicate with each other has been prevalent for many years. Various technologies and standards have been proposed and some are already in use today. Many of these technologies, both software and hardware, have paved the path for the adoption of WoT. The primary challenge is to address the *interoperability* of the tirade of things that surround us today. We present some of the early attempts that enabled communication of things on the Web and also look at technologies that are driving the research on the WoT.

Several industry alliances and standards have been defined like UPnP, DLNA, SLP, and Zeroconf to facilitate interoperability of real-world things. Each of these standards has individually been successful in enabling devices to communicate with each other. However, the availability of many standards creates an even more complicated challenge, which is *interoperability* between standards. With the diversity of things, their vendor specific properties, and the variety of services they provide, the use of standard-based interoperability approach is a never-ending process [7]. We briefly present two of these that have made considerable impact.

An early and noteworthy example is *Zeroconf* [8], which is a technique to provide reliable networking without any manual configuration and support services. Zeroconf is offered using several implementations. Probably the most known is Apple's *Bonjour* which is used to discover shared printers for streaming media to Apple TV devices.

Universal Plug and Play (UPnP¹) is a suite of networking protocols promoted by the UPnP forum and mainly used for devices to discover each other and establish connections over a common network. UPnP is based on protocols and standards like HTTP, HTTPU (i.e., HTTP over UDP), UDP, TCP, IP, etc. UPnP sup-

¹ <http://www.upnp.org/>

ports various services like discovery, description, control, event notification, and presentation. However, this cannot be widely used because in some embedded devices that are resource constrained it is impossible to host a large number of these protocols [9, 10]. Conventional middleware technologies such as CORBA and Java-RMI have been used to realize device interoperability by standardizing common protocols. The *Jini* project created a middleware where each device is loosely coupled as a Java object. Allard et al. introduced a framework to combine Jini with UPnP [11].

Technologies Driving the Web of Things

There are an increasing number of Internet-enabled things flooding the markets. Alliances such as the IP for Smart Objects (IPSO²) and the European Future Internet Initiative (EFII³) have accelerated this trend to connect a variety of physical things to the Internet, with the intention of propagating the wide use of Internet as the common medium for communication. Also, with the ever-decreasing size of embedded systems and related software footprint, it has become possible to directly integrate Web servers into many appliances [5]. This trend to use the Web as a platform to create and integrate applications that integrate real-world things into the Web has attracted attention in academia and industry [4, 6, 31]. The Service Oriented Architecture (SOA) has proven to be a promising technology for integrating devices into a business IT network [32]. This has in turn led to the propagation of using *Web services* to allow interoperability of things.

Web Services

The use of Web services is paramount in establishing interoperable distributed systems on a network. This indispensable technology is extended to WoT to allow data or functionalities of real-world things to be abstracted and exposed as services on the Web. There are two major classes that regulate the proliferation of Web services, the REST-compliant Web services and the arbitrary Web service protocols stack (WS-*) [12, 13].

RESTful Web Services

RESTful Web services are based on Representational State Transfer (REST) [13], an architectural style that considers every participating component to be an

² <http://www.ipso-alliance.org/>

³ <http://www.future-internet.eu/>

addressable *resource*. The state of each resource is essentially captured in a document referred to as a *representation*. The Web was built upon on this architecture where each resource is a Web page identified by a URI (Uniform Resource Identifier). REST is lightweight, simple, loosely coupled, flexible and seamlessly integrates into the Web using HTTP as an application protocol. RESTful Web services are based on the following four concepts [14].

- *Resource identification*: Resources are identified through URIs which provide a unique and global presence for service discovery.
- *Uniform interface*: Resources are accessed using the standard HTTP verbs, GET, POST, PUT, and DELETE.
- *Stateful interactions through hyperlinks*: Interactions with resources are stateless and self-contained. To circumvent this, state can be embedded into response messages to point to valid future states of the interaction.
- *Self-descriptive messages*: Resources are decoupled from their representation so that their content can be accessed in a variety of formats like Extensible Markup Language (XML), JavaScript Object Notation (JSON), plain text or YAML Ain't Markup Language (YAML).

Though REST has been used to build the traditional Web as we know it, these same attributes made REST an ideal platform to expose real-world thing services on the Web [27, 28]. IETF has constituted a working group called *Constrained RESTful Environments* (CoRE⁴), to develop a framework for resource oriented applications intended to run on constrained IP networks like WSN (Wireless Sensor Networks). This includes sensor based applications like temperature sensors, light switches, and power meters, applications to manage actuators like light switches, heating controllers, and door locks, and applications that manage devices. As part of the framework for building these applications, CoRE plans to define Constrained Application Protocol (CoAP) for the manipulation of resources on devices. The CoAP will be designed for interaction between devices on traditional IP networks, on constrained IP networks, and between constrained networks on the Internet. The CoAP is a work in progress but some implementations are already emerging such as the Firefox extension Copper⁵, Tiny OS⁶, and libcoap⁷. The CoRE group has proposed the following features for the CoAP:

- RESTful design minimizing the complexity of mapping with HTTP,
- Low header overhead and parsing complexity,
- URI and content-type support,
- Discovery of resources provided by CoAP services,
- Subscription for resources and resulting push notifications, and
- Caching mechanism.

⁴ <http://datatracker.ietf.org/wg/core/charter/>

⁵ <https://addons.mozilla.org/en-US/firefox/addon/copper-270430/>

⁶ <http://docs.tinyos.net/tinywiki/index.php/CoAP>

⁷ <http://libcoap.sourceforge.net/>

A REST-based approach was, also, used to define the TinyREST architecture [15], which introduces a new HTTP method, SUBSCRIBE, that enables clients to register their interests to device level services. The pREST or pico-REST [16] proposed by Drytkiewicz et al. is an access protocol, which brings the REST concept to the device level. The emphasis is on the abstraction of data and services as resources in sensor networks.

Though the work on REST-based services continues to encompass many WoT applications, in enterprise applications like banking or stock markets, where QoS (Quality of Service) levels are stricter, a more tightly coupled service paradigm like WS-* would be more ideal [14].

WS-* based Services

The key enabling technologies of WS-* are Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI). The services use HTTP as a transport protocol over which SOAP messages are sent. SOAP [17] is an XML-based protocol defining a message based architecture and format. The SOAP message is contained in an XML element called *envelop*, which includes two XML elements *header* and *body*. WSDL [18] is also an XML-based language describing the syntactic communication for message exchange between endpoints. The SOAP messages and sequences are abstractly described by WSDL. Service endpoints are addressed either on the transport level, i.e., with Universal Resource Identifiers (URIs) for SOAP/HTTP-bound endpoints, or on the messaging level via WS-Addressing [19]. UDDI [20] is also an XML-based registry framework for discovering Web services. While this technology has been mature and stable for several years now, it has failed to reach widespread acceptance in the industry.

WS-* is stateless but WS-Resource Framework [21] describes the interaction of Web services where the state needs to be maintained. The WS-* technology stack also covers many QoS features required to ensure the interoperability of distributed systems [22]. A large set of WS-* specifications has been suggested because, this approach enables the composition of Web services and is, also, flexible to be used by real-world things. Research on integrating WS-* into real-world things has been active for many years. A Service-Oriented Device Architecture (SODA) [23] is proposed to integrate many physical devices into enterprise systems. The architecture exposes all sensors and actuators as abstract business services. Pintus et al. [24] proposed an SOA framework where real-world things are described using WSDL and communications (between things) are modeled as service orchestrations using BPEL (Business Process Execution Language). The SOA approach for networks with embedded systems has also emerged in projects such as SIRENA (Service Infrastructure for Real-Time Embedded Networked

Applications) [25] and SOCRADES (Service Oriented Cross-layer Infrastructure for Distributed smart Embedded devices) [26].

Merging WS-* and RESTful Web Services

Since the goal of both paradigms is the same, there have been attempts of merging the two service architectures for WoT to make up for the disadvantages of each [27, 28]. The first direction is to have the RESTful architecture merged into the existing WS-* service architecture [27], which requires the implementation of alternate data model. Having two data models for the same service results in a very complex architecture and is not desirable. On the contrary the other direction is to have a software module that translates RESTful requests into WS-* requests [28]. This provides a structured approach ensuring a REST centric architecture while maintaining the robustness of WS-* approach. On the flip side, there is an effect on the performance of the RESTful APIs.

Embedded Web Servers

Embedded Web servers are an indispensable component for the long term adaptation and spread of WoT. These tiny Web servers enable communication between real-world things over the simple and widely used HTTP protocol. Researchers have successfully embedded tiny Web servers into resource-constrained things like sensors and smart cards, making Web-enabled things a reality [29, 30]. Filibeli et al. [33] implemented an embedded Web server based home network system where Ethernut⁸ based Web servers are embedded into home appliances. The home appliances are controlled and managed via the Web. Priyantha et al. [34] present an approach to implement a Web server on nodes of a sensor network using 15.8 KB ROM and less than 1KB RAM. Today, open source tiny Web server modules like FlyPort⁹ from OpenPicus are readily available. This module is 35X48 mm in dimension with an integrated 802.11 b/g/n Wi-Fi interface module and a 16-bit processor. The internal flash of 256KB has been demonstrated to be sufficient for different Web applications that access and control real-world things like switches and sensors. With the reducing physical size of hardware components and its software footprint many of the new appliances in the market, like Smart TVs and refrigerators are expected to come with embedded servers. It is very much possible that in the near future, embedded Web servers will be common feature in many of the physical things that surround us, enabling intelligent WoT applications.

⁸ <http://www.ethernut.de/>

⁹ <http://www.openpicus.com/>

Smart Things

Research and development on the IoT gained its recent momentum from the wide spread deployment and use of RFID (Radio Frequency IDentification). As illustrated in Fig.2, information of a *thing* like a can of beans, become available on the Internet when it is tagged with an RFID tag. The *RFID reader* reads the information on a tag and an *Application server* uses the reader's access protocol to access the information of the tags and expose the information on the Internet.

It is important to clarify the term *things* while considering WoT or IoT applications and systems. While the term *devices* or *appliances* are often used to refer to things under consideration, some tend to believe that almost every “thing” can be included. The rationale of the latter case is that information of any *tagged thing* is accessible on the Internet while the former case argues that only the *connected devices* on the Internet fall into the category of *things* in IoT. Today, there are RFID readers that are Internet enabled which directly access the Internet without the use of an intermediate server and has been elevated into the category of *connected devices* or *connected things*.

Fig.2. Internet enabling things with RFID

With the advances in technology the capabilities of real-world things have been increasing and this enhancement has introduced the term *Smart* to be appended into these things. Thus, we have Smart phones, Smart TVs, and Smart Homes. The proliferation of *Smart* things with Web capabilities is driving the possibilities of realizing the WoT. Kortuem et al. [35] address issues on modeling and representing *Smart* objects and also striking a balance between the objects and the infrastructure. They classify Smart thing as: Activity-aware objects, policy-aware objects, and process-aware objects. These types represent combinations of three dimensions with the aim to highlight the interdependence between design decisions and explore how these objects can cooperate to form IoT. Mathew et al. [36] proposed an ontological approach to classify and manage physical things within ambient spaces where the requirement of additional capabilities uses an ontology,

which recognizes four fundamental dimensions of candidate elements. These dimensions of *Identity*, *Processing*, *Communication*, and *Storage* are referred to as the IPCS set. A taxonomy of things facilitate the process by which things are made *Smart* by augmenting their IPCS capabilities.

6LoWPAN

Interoperability of things would only be possible on a scalable, accessible, and end-to-end communication infrastructure. To enable embedded Web server on devices, they must first be connected to the Internet (IPv6) (Fig.1.). Things with inherent information, that has to participate in a Web based application, are of various types and capabilities. The IP protocol stack should be adapted for devices with limited capabilities. *6LoWPAN* (IPv6 over Low power Wireless Personal Area Networks) was launched by IETF which defines mechanisms that allow IPv6 packets to be sent to and received between resource constrained devices usually by adopting low-power radio communication protocols like IEEE 802.15.4.

Fig. 3. Comparison between TCP/IP and 6LoWPAN protocol stacks

Zeng et al. [37] notes that in 6LoWPAN, the IP communication is provided above an adaptation layer and the necessity is mainly to fit one IP packet within one 802.15.4 MAC (Layer 2) frame. Fig.3 shows a layer-wise comparison between the TCP/IP and the 6LoWPAN stacks. The adaptation layer enables IPv6 packets to fit into IEEE 802.15.4 frame payload. It also manages header compression, packet fragmentation and reassembling and edge routing. 6LoWPAN is ideally implemented using UDP as transport protocol which ensures performance and efficiency when dealing with Internet-enabled things. Web applications will use HTTP over UDP and hence will be robust due to the unreliability of UDP. As the number of things on the Internet increases 6LoWPAN makes them addressable at the IP layer.

Architecture for the Web of Things

Architecture for the WoT would have to primarily take into consideration the methods for integrating the plethora of things on the Web and then formulate an abstraction framework that would make real-world things accessible and interoperable. We look at these aspects with reference to the ongoing efforts to realize architecture for the WoT.

Integrating Things on the Web

As illustrated in Fig. 1 (b), the requirement of Web-enabling a thing is to expose its operations as HTTP URLs. We explain two ways this can be achieved, (1) Direct Integration: augmenting the Web server to the thing, or (2) Indirect Integration: expose operations of a thing through a proxy Web server [6].

Direct Integration: Augmenting a Web server

A thing is Internet-enabled and Web-enabled by augmenting it with the capability to communicate over TCP/IP or HTTP respectively. An embedded Web server exposes the thing's operations through the use of URLs and also gives it an IP address. When using RESTful APIs, a thing's operations are exposed through standard Web operations like GET or POST. Ostermaier et al. [5] present a prototype using Wi-Fi modules for connecting things directly to the Web. They enabled association of sensors, actuators, and things with each other through the Web. Guinard and Trifa [6] present the direct integration of *Sun SPOT*¹⁰ with a Web server. They implement the embedded server directly on the Sun SPOTs nodes. The server natively supports the four main operations of the HTTP protocol i.e., GET, POST, PUT, and DELETE, making the Sun SPOT Web-enabled. Akribopoulos et al. [38] introduce a Web service-oriented architecture for integrating small programmable objects in the WoT. Using Sun SPOT applications and sensor data exposed through Web services, they present two use cases for building automation and remote monitoring.

Indirect Integration: Using a Proxy Web server

The process of indirect integration involves a gateway or proxy server between the thing and the Web. A thing could have potential information that needs to be made available on the Web but does not have the necessary capabilities to com-

¹⁰ <http://www.sunspotworld.com/>

municate over TCP/IP or HTTP. The minimum requirement for such things to be an eligible candidate for the WoT is that they must be uniquely identifiable within a particular context. Mathew et al. [36] refer to such a thing as a *Core* thing and examples of such things would be pallets, medicine bottles, or shoes, which can be identified uniquely on the Web using an identification system like RFID or Barcode (Fig.2.). The information of *Core* things are made available to the Web through the proxy or gateway server, abstracting the proprietary protocol with uniform Web-based APIs.

Trifa et al. [39] implement a gateway for Web based interaction and management of sensor networks through RESTful Web services. Guinard et al. [40] build an EPC Network prototype by using virtualization, cloud computing, and Web technologies. In their prototype, the RFID reader behaves like a gateway which locates between the cloud server and RFID tags. Welbourne et al. [41] develop a suite of Web-based tools and applications that facilitate the understanding, management, and control of personal RFID data and privacy settings. They deployed these applications in an RFID ecosystem and conducted a study to measure the trends in adoption and utilization of the tools and applications as well as users' reactions.

Frameworks for the Web of Things

The use of Web services (WS-* and RESTful) is fundamental to any WoT architecture and various experiments have been tested and deployed. We look at two such architectures [26] and [28] to understand the layers used to expose things on the Web.

SOCRADES Integration Architecture (SIA)

The SOCRATES Integration Architecture (SIA) [26, 42] describes the use of both WS-* and RESTful services to access devices from enterprise applications.

The architecture includes a *local/on-premise subsystem* and a *central or remote or cloud subsystem*. A high-level illustration of the main layers of SIA with the modules of the local subsystem which interfaces with the devices is shown in Fig.4. SIA enables the integration of services of real-world things running on embedded devices with enterprise services. WS-* Web service standards constitute the main communication method used by the components of enterprise-level applications. This enables business applications to access real-time data from a range of devices through abstract interface based on Web services. SIA, also, supports RESTful services to be able to communicate with emerging Web 2.0 services. This enables any networked device that is connected to the SIA to directly participate in business processes.

Fig.4. Main layers of SOCRADES Integration Architecture (SIA)

The *Local/On Premise subsystem* features a *Local Discovery Unit (LDU)* connected to devices seen on a LAN, and the *Central subsystem* (anywhere on the network) hosts enterprise-level applications. The LDU module scans the local network for devices and reports their connecting and disconnecting to the central system. It acts as an intermediary that provides uniform access to different classes of devices through a set of platform-dependent plugins. In the local subsystem at the *Devices Layer* there are several embedded devices that run various services. The legacy devices would require a *Gateway* to expose their proprietary functionalities.

SIA is able to interact with devices using several communication protocols, such as DPWS (Device Profile for Web Services), OPC-UA, and REST. The details of each layer and their functionalities are provided in [26].

The Web of Things Application Architecture

The WoT application architecture presented by Guinard [28] enables the integration of things with services on the Web and facilitates the creation of Web based applications that operate on real-world things. The primary goals of the architecture is to enable rapid prototyping of applications that integrate real-world things for developers, to offer direct Web based access to things to Web users, and to offer lightweight access to things data which would enable the data to be consumed by other things or software.

Fig.5. Application Architecture for the Web of Things

In the proposed layered architecture, each layer does not hide the lower layers but rather provides a hierarchy of separate abstractions to connect things as first class citizens of the Web [28]. Based on requirements, an application can be built on top of each layer or a combination of layers. Fig.5 illustrates the layers and also their roles, we present a summary of these layers here; a detailed representation is given in [28].

The *Accessibility* layer focuses on providing a standard access to all kinds of connected things. This layer exposes things as *RESTful Things* using resource oriented architecture. REST [13] is used as a universal technique to interact with real-world things and four steps are suggested to Web-enable things:

1. Design the resources by identifying the functionality or services of a thing and organizing their interactions,
2. Design the representations and decide on how these will be provided,
3. Design the interfaces which indicate the actions of each service, and
4. Decide on direct or indirect integration into the Web.

Guinard [28] indicates that a client-initiated interaction model, where a Web client continuously polls a Web-enabled thing, is costly when dealing with resource constrained things. Instead a server-initiated model is proposed. Here, a Web client first registers with a Web server and is notified when an event occurs. This creates a real-time approach when a Web client interacts with a Web-enabled thing hosting a Web server. Moreover, the use of *tPusher*, which adds the use of *WebSockets*¹¹ to the WoT architecture, is suggested. *tPusher* moves the protocol to a gateway rather than a real-world thing [40]. This layer ensures that real-world things are exposed as first-class citizens on the Web. The *Findability* layer focuses

¹¹ <http://dev.w3.org/html5/websockets/>

on making services of real-world things searchable and available for service consumption or composition. Integration of things to the existing search engines with the use of semantic annotations is studied and the shortcomings of this approach were presented. The shortcomings are because of the inherent nature of real-world things, i.e., real-world things do not have properties that can be indexed like documents on the Web and things are tightly coupled with their contextual information like the owner of a thing, or its location. A lookup and registration infrastructure adapted to WoT is proposed. The *Sharing* layer focuses on how Web-enabled things are shared ensuring privacy and security. The requirements stated for a sharing platform for WoT are security, ease of use, reflection of trust models, interoperability and integrated advertisement. The architecture proposes the use of existing social networking platforms like Facebook, Twitter, and LinkedIn, as sharing hubs for things. Since these platforms already manage access control and trust measures, these can be leveraged to manage the access and privacy of things privately. The *Composition* layer focuses on enabling end-users to create simple composite applications with Web-enabled things. Web 2.0 mashup techniques are reused to create Physical mashup editors considering the requirements of the real-world things. The requirements are, support for event-based mashup, support for dynamic building blocks, support for non-desktop platform and support for application specific editors [43].

Building WoT Applications

Existing tools, techniques, languages and models for building Web applications can be applied to the WoT. However, for applications on resource constrained things where ad-hoc and event driven interactions are necessary, the challenges need to be understood and some techniques need to be adopted. We present some of these key techniques here.

The AJAX Technique

The AJAX (Asynchronous JavaScript and XML) technique [44] for Web application development creates highly interactive and efficient applications with efficient workload distribution between the client and the server modules. The behavior of an AJAX Web application can be separated into two phases: loading phase and running phase [29]. In the loading phase the client collects files that contain style, content, and application code, while in the running phase, the client executes the application code from the first phase, and interacts with the server by sending asynchronous requests, allowing the Web page to update itself dynamically. Experimental results [29] show that during the first phase, numerous large-sized content are served and during the second phase smaller size content is sent to the cli-

ent. The AJAX technique uses the browser to handle the workload and relieves the Web server. This is an interesting technique for the Web of Things applications, where the embedded Web servers have lesser resources when compared to the clients.

Mashups - Composing Things on the Web

Mashups have eased the process of developing WoT applications. This new technique introduces a paradigm shift in the creation of Web applications by using Web 2.0 techniques and composing various Web services to achieve the application goals. Unlike portal-based solutions, the content aggregation for mashup applications can occur either on the client or on the server. This is beneficial for the WoT applications, where all the real-world things are abstracted as Web resources, which are addressable, searchable, and accessible on the Web. Guinard et al. [6], present two representative mashup styles, physical-virtual mashups and physical-physical mashups. For the former, the *Energy Visualizer* application was developed which has a Web interface to monitor power consumption and to control different home appliances. For the latter, to present the composition of services offered by things, the *Ambient Meter* was implemented on Sun SPOTs, which polls predefined URLs to get the energy consumption of devices in a room using HTTP based requests and responses. We discuss the Ambient Meter case study later in the chapter.

Event Driven Approach

The limitation of resources on embedded Web servers is a challenge that needs to be considered when creating applications for the WoT. Hence, a feature that is important to WoT application is the need to push data from server to client, based on events instead of the client continuously requesting to pull data from server. Event driven approaches are efficient to implement stateless behaviors and hence software designed for embedded systems ideally use event-driven approaches to manage resource constrained hardware.

Comet [45] allows a Web application to push data from the Web server to the client. Comet¹² is an umbrella term for techniques which are used for this interaction and implementation methods are of two types: *Streaming* and *Long Polling*. In *Long Polling*, the client polls the server for an event by sending requests and stops only when the server is ready to send some data. After it gets a response the client will restart a Long Polling request for the next event. In *Streaming*, the cli-

¹² [http://en.wikipedia.org/wiki/Comet_\(programming\)/](http://en.wikipedia.org/wiki/Comet_(programming))

ent opens a persistent connection from the browser to the server for all events. Neither side closes the connection and in the absence of events servers periodically send dummy data to keep the connection active. While these workaround methods are in practice they pose two drawbacks, they generate unnecessary traffic and they are extremely resource demanding on the Web servers. While Comet is better in terms of data consistency and managing traffic workload, it has scalability issues.

More recently, WebSockets were proposed which use full duplex communication with a single TCP/IP socket connection. It is accessible from any compliant browser using simple JavaScript API. The increasing support for HTML5 in Web and Mobile Web browsers makes it a very good candidate for pushing data for the WoT. Since WebSockets protocol has an initial handshake followed by message framing, layered over TCP, they can be implemented in a straightforward manner on any platform supporting TCP/IP [40].

Case Studies of Ambient Environments

As stated earlier, the greater vision for the WoT is the creation of ambient spaces where people and things seamlessly communicate to achieve a sustainable and efficient environment. We present two case studies that use WoT concepts to create and realize ambient spaces.

Ambient Classrooms

Fig.6. Classrooms abstracted as ambient spaces on a campus Intranet

Campus communities are notorious for their huge energy consumption despite awareness campaigns to reduce this consumption [46]. With the intention of creating sustainable campus environments we propose a framework which abstracts physical units like classrooms into *Ambient Classrooms*. The framework makes these classrooms aware of the energy footprint of the users that reserved the room and also manage the usage of things like lights and projectors. The framework illustrated in Fig. 6, adopts a service-oriented approach to connect classrooms as *Ambient Space* (AS) units on a campus. All classrooms on campus have similar things like, projectors, lights, and cameras. Each AS unit has a *Space Manager* comprised of hardware and software modules that enable things to communicate on the Web. The *Controller* provides Web services that grant operation of things within the AS. Things in the AS that are not Web enabled are augmented with Web capabilities using *Adapters* to provide the desired functionality. The *Rules Engine* constrains thing's usage based on operational intelligence which are specific to different types of physical things in AS. The *Monitor* provides interfaces with sensors that capture the status of AS for various measurements. Each AS has a designated set of repositories which is part of the campus *Data Center*. The *Space Repository* holds details like, location, purpose, and seating capacity, of all AS on campus. The *Things Repository* forms a knowledge base of all things associated with each AS and the *Service Repository* provides a directory of services exposed by each AS. The functionalities for the Space Manager are provided through two subsystems, the *Things Control and Sensing* (TCS) subsystem and the *Classroom Reservation and Control* (CRC) subsystem. We consider *lights* in the classrooms as candidates of Web-enabled things for this discussion.

The TCS subsystem comprises of Web-enabled things that are either augmented with embedded Web servers or indirectly connected to a proxy Web server. The TCS provides direct Web based access (RESTful APIs) to things like light bulbs in the room and is designed as the last unit of information flow as it may have the least computing resources. Lights have two basic operations, *on* and *off*. The challenge is to Web-enable inanimate things like light bulbs in order to push and pull information. Their capabilities are to be enhanced so that they can be controlled, sensed and their operations are accessible over the Intranet. We recently proposed an ontological approach to classify physical things within an AS environment and manage them from the Web [36]. The taxonomy of things facilitates the process by which things are made *Smart* by augmenting their IPCS capabilities, as mentioned earlier. For enabling *Smart Lights*, these capabilities are satisfied by:

- Identifying lights uniquely over the Intranet (IP address) and the Web (URL).
- Augmenting processing capability so that Web requests are processed and relevant actions are converted into electrical operations.

- Enabling communication capabilities to lights so that user actions and sensing results are communicated to and from the Smart lights.
- Providing storage capabilities for the lights to cache Web resources and status information.

This is realized by augmenting the classroom lights with tiny Web servers¹³ and light sensors. TCS exposes two *Smart Light* services using RESTful expressions, *Light-ControlWS* to switch lights on or off and *Light-SenseWS* to detect the status of lights.

The CRC subsystem manages scheduling functions for the Ambient Classrooms and also maintains the rules for controlling things. Interactions with the Ambient Classrooms are provided through the CRC implemented on a digital classroom signage. The signage displays details of ongoing sessions and uniquely identifies a classroom. It is connected to the Intranet and accessed both online and onsite using a touch screen interface. The CRC hosts a *Calendar Service* that allows the scheduling of classrooms for courses. A genuine user with a valid username and password or a Web application like a time tabling application can reserve classrooms through CRC based on available time slots. CRC also hosts the *Ambience Service* to manage the state of things inside the classrooms depending on preset rules.

Rules are set to turn lights on or off based on room usage and also to monitor lights to determine if any of the lights need to be replaced. Classroom users are made aware of their energy usage based on the duration of time they use the classrooms. Each user monitors their individual *Energy Account* through an online Web portal, and incentives for optimal usage are planned.

The proposed framework was evaluated and it was noticed to save considerable amount of energy when compared to earlier usage scenarios.

Ambient Meter

The Ambient Meter is a prototype implementation that demonstrates the composition of real-world services of things using RESTful APIs. It specifies how things can interoperate to create new systems over HTTP. Guinard and Trifa [6] deployed this system on a mobile device that monitors the energy usage of all appliances in its vicinity. The Ambient Meter is sensitive to its location and it automatically adapts to the place it monitors, without human intervention. Based on the amount of energy consumption in the room, the Ambient Meter changes color to display the variations in energy footprint, from very green (energy consumed is low) to very red (energy consumed is high). Each room has various *Ploggs*¹⁴ that

¹³ <http://www.openpicus.com/>

¹⁴ <http://www.plogginternational.com/>

are connected to appliances, like kettle, lamp, and PC. Their energy consumption is measured and processed by a *Smart Gateway*, which communicates with the Ploggs. A Smart gateway is co-located with the Ploggs in a room, which discovers the Ploggs and records the total energy consumed. The Smart Gateway is deployed alongside with an LLDU (Local Lookup and Discovery Unit) for resolving current location [6, 28]. When the mobile Ambient Meter (RESTful Sun SPOT) is located in a room it connects to the gateway in that room and becomes part of the local network. As the meter polls the gateway using a standard URI, a JSON representation of the energy consumption of all the Ploggs in the room is received. When the meter moves to another location it connects to the gateway in that room but uses the same URI to get the energy consumption recorded by the gateway in the room. Using RESTful approach the integration of the devices were reduced to building a Web mashup, where all the services are invoked by means of simple HTTP requests.

Research Focus and Challenges

The WoT is transforming IoT technology just as the Web has transformed computing over the last decades. WoT enabling techniques are maturing and creating new platforms for designing and realizing innovative applications. We discuss some of these challenges and ongoing research towards the realization of the WoT vision.

Research has been focusing on adapting traditional client-server Web approaches to expose the functionalities of physical things [28]. This works fine when clients initiate interactions with servers to pull data and services from embedded applications that are expected to control things on the WoT. However to monitor things in ambient environments, the required applications are often event-based and hence, Web-enabled things should also be able to push data to clients. Hence, standards such as HTML5¹⁵ are moving towards asynchronous bi-directional communication to enable such interactions, where the server initiates events [28].

The spread of WoT is expected to flood the Web with real-time and real-world information and services that need to be discovered [47]. The use of sensors and other probing devices result in the dissemination of a large amount of real-time information. In contrast to the documents on the traditional Web, real-world things on the Web are expected to rapidly generate dynamic content because of their changing state. WoT applications inherently depend on the context of things and hence, search engines for WoT should focus on efficiently probing real-time data and discovering dynamic services. The efficiency and performance issues of search engines and related algorithms form some of the current areas of WoT re-

¹⁵ <http://dev.w3.org/html5/eventsources/>

search. Recently, Mayer et al. [48] suggested a discovery service for Web-enabled things, where users employ RESTful interfaces for discovering Web resources. A key challenge is to have an efficient search engine, which is able to reach and retrieve properties of real-world things on the Web. We presented some of these properties in light of the dynamic nature of real-world things [49].

Research is also directing focus towards optimizing communication protocols for resource constrained devices. Efforts to adapt application layer protocols like CoAP (Constrained Application Protocol) [50] based on REST and HTTP are active research areas, to enable flexibility and ease of use for a Web user and not just technical people. Refining protocols like 6LoWPAN [50, 51] that adapt the IPv6 protocols for low power radio networks is, also, highly relevant.

A key research challenge is the study of smart things composition to create context-aware ambient environments [52]. With the plethora of real-world things that can be participants on the Web, this challenge is sophisticated and many innovative solutions are yet to be proposed. Research is focusing on the study of consumer reaction and trends in applications for eHealth, smart homes [53, 54], and sustainable environments [55], which are being suggested using ambient intelligence.

Issues of security, privacy, and trust in WoT promise many opportunities for innovations [56]. The use of REST-based interfaces makes it possible to have secure interactions using HTTP authentication or HTTPS [57]. As things on the Web will be accessible and shared among many users, research in this area is crucial to the success and widespread use of the WoT. Research has been rampant with innovative ideas dealing with these issues in the IoT [58, 59], but the focus on WoT is yet to mature. To illustrate these developments, consider the example of a car on the Web to describe these issues with respect to WoT. *Privacy* would involve dealing with issues that arise when the current owner decides to share the car with others on the Web. *Security* would deal with issues that pertain to who or what will have access to the car when it is in use. *Trust* would deal with issues of interactions between things on the Web, like if the garage door would open when the car arrives. The use of the social Web as a platform to ensure the privacy of things has been advocated [60], to control Web-enabled things among trusted members on social Web sites.

Conclusion

The Web of Things (WoT) is a rapidly expanding area of research and development with the potential to realize Weiser's vision. Salient technologies are driving the success of the WoT, such as embedded systems, Web services, and smart things. In this chapter, we have presented and discussed existing technologies and platforms that support the WoT. We presented different approaches to Web-enable

things to facilitate the realization of WoT based applications. The approaches are motivated by the level of comfort that people have when dealing with the Web, to portray the WoT as a simple and do-it-yourself technology. Advances in communication and real-time systems indicate that real-world things will soon become IP-enabled and embed Web servers, making them addressable, controllable and composable on the Web. In this context, we surveyed ongoing research, which addresses some of the challenges posed in leveraging and adapting existing technologies for pushing and pulling information from real-world things. Many innovative ideas and applications are being designed and deployed on WoT creating ambient environments where people and real-world things seamlessly interact over the Web. We discussed two case studies to illustrate developments to realize ambient environments. Some ongoing research trends and challenges were also presented, which are presently driving the WoT into new horizons.

References

- [1] M. Weiser. *The Computer for the 21st Century*. Scientific American, 265(3):94–104, 1991
- [2] G. Mulligan. *The Internet of Things: Here Now and Coming Soon*, IEEE Internet Computing, 14(1):35-36, 2010
- [3] T. Berners-Lee, *The Web of Things*, Special theme on the future web, ERCIM News – the European Research Consortium for Informatics and Mathematics, 2008. Retrieved on 2012.02.12 from <http://ercim-news.ercim.eu/en72/keynote>
- [4] D. Raggett. *The Web of Things: Extending the Web into the Real World*, SOFSEM 2010: Theory and Practice of Computer Science, Jan 2010
- [5] B. Ostermaier, M. Kovatsch, and S. Santini, *Connecting Things to the Web using Programmable Low-power WiFi Modules*, In the proceedings of the 2nd International Workshop on the Web of Things (WoT 2011), 2011
- [6] D. Guinard and V. Trifa, *Towards the Web of Things: Web Mashups for Embedded Devices*, In the proceedings of the workshop Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM'09), 2009
- [7] Z. Song, A. A. Cardenas, and R. Masuoka, *Semantic Middleware for the Internet of Things*, In the proceedings of the IEEE. Internet of Things (IOT), IEEEPress, pp. 1-8, 2010
- [8] S. Cheshire, and D.H. Steinberg, *Zero Configuration Networking, the Definitive Guide*, O'Reilly, 2005
- [9] S. Duquennoy, G. Grimaud, and J.J. Vandewalle, *Smews: Smart and Mobile Embedded Web Server*, International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '09), pp. 571 – 576, 2009
- [10] H. Song, D. Kim, K. Lee, and J.Sung, *UPnP-Based Sensor Network Management Architecture*, In the proceedings of the International Conference on Mobile Computing and Ubiquitous Networking, 2005
- [11] J. Allard, V. Chinta, S. Gundala, and G. Richard, *Jini meets UPnP: An architecture for Jini/UPnP interoperability*, In the proceedings of the Application and the Internet, 2003
- [12] W3C Working Group, *Web Services Architecture*, <http://www.w3.org/tr/ws-arch>
- [13] R. T. Fielding, *Architectural styles and the design of network-based software architectures*, Ph.D. dissertation, 2000
- [14] C. Pautasso, O. Zimmermann, and F. Leymann, *Restful web services vs. 'big' web services: making the right architectural decision*, In the proceedings of the 17th international conference on World Wide Web, ser. WWW '08. New York, NY, USA: ACM, pp. 805–814, 2008

- [15] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, *TinyREST: A protocol for integrating sensor networks into the Internet*, In the proceedings of the REALWSN, 2005
- [16] W. Drytkiewicz, I. Radusch, S. Arbanowski, and R. Popescu-Zeletin, *pREST: a REST-based protocol for pervasive systems*, in Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on. IEEE, pp.340–348, 2004
- [17] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer, *Simple object access protocol (SOAP) 1.1*, 2000
- [18] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, *Web services description language (WSDL) 1.1*, 2001
- [19] W3C, *Web Services Addressing*, <http://www.w3.org/2002/ws/addr/>
- [20] T. Bellwood, S. Capell, L. Clement, J. Colgrave, et al., *UDDI Version 3.0.2*, Oasis, <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>
- [21] OASIS. *Web Services Resources Framework (WSRF 1.2)*, 2006, <http://www.oasis-open.org/committees/wsrfl/>
- [22] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson, *Web Services Platform Architecture*, Prentice Hall, 2005
- [23] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, *SODA: Service oriented device architecture*, Pervasive Computing, IEEE, vol. 5, no. 3, pp. 94 –96, 2006
- [24] A. Pintus, D. Carboni, A. Piras, and A. Giordano, *Connecting smart things through web services orchestrations*, In the proceedings of the 10th international conference on Current trends in web engineering, (ICWE'10), Springer-Verlag, 2010, pp. 431–441
- [25] F. Jammes and H. Smit, *Service-oriented paradigms in industrial automation*, Industrial Informatics, IEEE Transactions on, vol. 1, no. 1, pp. 62 – 70, 2005.
- [26] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. S. d. Souza, and V. Trifa, *SOA-based integration of the internet of things in enterprise services*, In the proceedings of the 2009 IEEE International Conference on Web Services, (ICWS '09), IEEE Computer Society, pp. 968–975, 2009
- [27] D. Guinard, M. Mueller, and V. Trifa, *RESTifying Real-World Systems: a Practical Case Study in RFID*, Springer, 2011
- [28] D. Guinard, *A Web of Things Application Architecture - Integrating the Real-World into the Web*, Ph.D. dissertation, 2011
- [29] S. Duquenooy, G. Grimaud, and J.J. Vandewalle, *The Web of Things: Interconnecting Devices with High Usability and Performance*, In the proceedings of the International conference on Embedded Software and Systems (ICESS'09), 2009
- [30] I. Agranat, *Engineering web technologies for embedded applications*, Internet Computing, IEEE, vol. 2, no. 3, pp. 40 –45, 1998
- [31] *The Web of Things*, White Paper, Tridium Inc., Sep. 2009
- [32] D. Lizcano, M. Jime'nez, J. Soriano, J.M. Cantera, M. Reyes, J.J. Hierro, F. Garijo, and N. Tsouroulas, *Leveraging the Upcoming Internet of Services Through an Open User-Service Front-End Framework*, In the proceedings of the First European Conf. Towards a Service-Based Internet (ServiceWave '08), pp. 147-158, 2008
- [33] M. Can Filibeli, O. Ozkasap, and M. Reha Civanlar, *Embedded Web server-based home appliance networks*, Journal of Network and. Computer Applications, vol. 30, pp. 499–514, 2007
- [34] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, *Tiny web services: design and implementation of interoperable and evolvable sensor networks*, In the proceedings of the 6th ACM conference on Embedded network sensor systems, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 253–266.
- [35] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, *Smart objects as building blocks for the internet of things*, IEEE Internet Computing pp. 30–37, January/February 2010.
- [36] S.S. Mathew, Y. Atif, Q.Z. Sheng, and Z. Maamar, *Managing things in an Ambient Space*, In the proceedings of the 9th International Conference on Service Oriented Computing (ICSOC), PhD Symposium, Paphos, Cyprus, 2011.

- [37] D. Zeng, S. Guo and Z. Cheng, *The Web of Things: A Survey*, Journal of Communications, Vol 6, No 6 (2011), 424-438, Sep 2011
- [38] O. Akribopoulos, I. Chatzigiannakis, C. Koninis, and E. Theodoridis, *A Web Services oriented Architecture for Integrating Small Programmable Objects in the Web of Things*, Developments in E-systems Engineering, pp. 70–75, 2010
- [39] V. Trifa, S. Wiel, D. Guinard, and T. Bohnert, *Design and implementation of a gateway for web-based interaction and management of embedded devices*, In the proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE), 2009.
- [40] D. Guinard, C. Floerkemeier, and S. Sarma, *Cloud computing, rest and mashups to simplify RFID application development and deployment*, In the proceedings of the 2nd International Workshop on the Web of Things (WoT 2011). San Fransisco, USA: ACM, June 2011.
- [41] Welbourne, E. et. al. *Building the Internet of Things Using RFID: The RFID Ecosystem Experience*, IEEE Internet Computing, May 2009.
- [42] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, *Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services*, IEEE Transactions on Services Computing, 3, 223–235, 2010
- [43] D. Guinard, *Mashing up your web-enabled home*, In the proceedings of the 10th International conference on Current trends in Web Engineering (ICWE '10), pages 442-446, Vienna, Austria, Springer-Verlag, 2010.
- [44] J.J.Garrett, *Ajax: A new approach to Web applications*, Adaptivepath, 2005
- [45] A. Russell, *Comet: Low Latency Data For Browsers*, The Dojo Toolkit, 2006
- [46] Campus Conservation Nationals: *Competition to Slash Energy Use at US Colleges*, Alliance to Save Energy, Nov 10, 2011, <http://ase.org/efficiencynews/campus-competition-slash-energy-use>
- [47] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmaier, and W. Kellerer, *A Real-Time Search Engine for the Web of Things*, In the proceedings of the International Conference on Internet of Things (IoT 2010), Japan, 2010
- [48] S. Mayer and D. Guinard, *An extensible discovery service for smart things*, In the proceedings of the 2nd International Workshop on the Web of Things (WoT 2011), USA, ACM, 2011.
- [49] S.S. Mathew, Y. Atif, Q.Z. Sheng, and Z. Maamar, *Ambient things on the Web*. Journal of Ubiquitous Systems and Pervasive Networks (JUSPN) 1(1), 1–8 (2010)
- [50] Z. Shelby, *Embedded Web services*, IEEE Wireless Communications, 17(6):52-57, December 2010
- [51] G. Mulligan, *The 6LoWPAN architecture*, In the proceedings of the fourth workshop on Embedded networked sensors (EmNets '07), 78-82, Ireland, ACM, 2007
- [52] M. Boussard, B. Christophe, O. Le Berre, and V. Toubiana, *Providing user support in Web-of-Things enabled Smart Spaces*, In the proceedings of the Second International Workshop on Web of Things (WoT '11). ACM, USA, 2011
- [53] L. Compagna, P. El Khoury, F. Massacci and A. Saidane, *A Dynamic Security Framework for Ambient Intelligent Systems: A Smart-Home Based eHealth Application*, Transactions on Computational Science X, Lecture Notes in Computer Science, Springer, 2010
- [54] T. Mantoro, M.A. Ayu, and E.E. Elnour, *Web-enabled smart home using wireless node infrastructure*, In the proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia (MoMM '11). ACM, USA, 2011
- [55] A. Paetz, E. Dütschke and W. Fichtner, *Smart Homes as a Means to Sustainable Energy Consumption: A Study of Consumer Perceptions*, From Knowledge to Action - New Paths towards Sustainable Consumption, Journal of Consumer Policy, Springer, 2012
- [56] V. Gupta, P. Udupi, and A. Poursahi, *Early lessons from building Sensor.Network: an open data exchange for the web of things*, In the proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010
- [57] E. Wilde, *Putting Things to REST*, UCB iSchool Report 2007-015, School of Information, UC Berkeley, 2007

- [58] R. H. Weber, *Internet of Things - New security and privacy challenges*, Computer Law & Security Review 26 (1), 2010
- [59] C. M. Medaglia, and A. Serbanati, *An overview of privacy and security issues in the internet of things*, The Internet of Things, 389-395. Springer, New York, 2010
- [60] D. Guinard, M. Fischer, and V. Trifa, *Sharing Using Social Networks in a Composable Web of Things*, In the proceedings of the 1st IEEE International Workshop on the Web of Things (WoT 2010) at IEEE PerCom, Germany, 2010