A Distributed Resource Management Architecture for Interconnecting Web-of-Things using uBox

Naoya Namatame
namachan@ht.sfc.keio.ac.jp

Till Riedel
reidel@teco.edu

Takashi Miyaki
miyaki@teco.edu

Yong Ding
ding@teco.edu

Hideyuki Tokuda
hxt@ht.sfc.keio.ac.jp

Michael Beigl
michael@teco.edu

ABSTRACT

Although there are many smart devices and networked embedded object applications using World Wide Web technologies, it is still a big step to go towards a true Web of Things. It is e.g. difficult to build ubiquitous WoT applications that work in and accross multiple environments. Approaches which aggregate WoT ressources by centralizing all the resource information, have problems: total dependency on external infrasture, lack of private WoT management, inflexible communication patterns and limited dynamic ressource discovery and mapping. To solve these problems, we propose *uBox*, a local WoT platform which can be a stand-alone server to make your WoT environment, with interfaces to connect the other local WoT platforms. This way, which we call *uBoXing*, we can create World Wide WoT platform with a distributed architecture. This paper describes the concept of a distributed resource management architecture, and how we implement the concept into software. Also, we will discuss the platform with the example application in SmartTecO environment.

1. INTRODUCTION

With the IPv4 Internet age at an end and IPv6 at the doorstep, the Internet of Things vision is gaining a sudden momentum. However, 340 Million IP addresses per capita alone do not solve the the problem of making ubiquitous computing possible. Ubiquitous computing is less about global network connectivity than new ways of interacting with physical reality. Therefore it is important to think more about what the Web of Things (WoT), the application layer of an Internet of Things. In this paper we show how WoT applications and the application level communication structure are strongly connected and what additional infrastructure are needed to enable novel ubquitous application using World Wide Web technology.

We present different aspects that are not sufficiently addressed by current WWW technology and that will have a direct effect on WoT application. Especially the dynamic aspect of ubiquitous computing infrastructures are often not covered sufficiently by the current WoT infrastructure. Making mobility, proximity and contextual relations between things explicit is a challenge for a purely technical Internet of Things. For example, a mobile life logging application in a smart phone that collects sensor data from the sensors around the user should acquire sensor data from several WoT domains, as shown in Figure1. This type of application is difficult to built on current WoT, since it collaborates with multiple domains of WoTs.

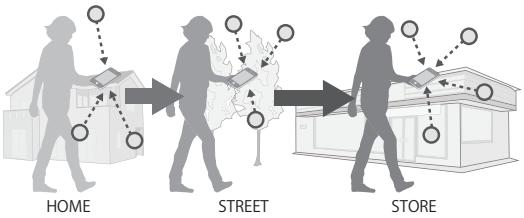

Figure 1: An application scenario that our platform enables. The application is collaborating with different domain of WoTs.

We show by introducing an active component into the WoT (the *uBox*) we can efficiently enable novel application types in the WoT without changing the application layer. The *uBox* is based on a light-weight application layer middleware that enables room-scale ubiquitous computing, mobile scenarious and Internet-scale interaction by building a decentralized infrastructures that intergrates into a World Wide WoT via *uBoXing*. *uBoXing* interconnects *uBoxes* and allows Internet scale ressource management of WoT devices. In the paper, we will describe our first lessons implementing a number of practical ubicomp applications, that are motivated and supported by the concept of the World Wide WoT platform.

2. RELATED WORK AND ANALYSIS

As we motivated in the introduction we hit principle barriers that anybody willing to build ubiquitous applications on the current infrastructure will quickly face. In the following section we analyze the reason why the WoT needs more that World Wide Web can offer today and why it still makes sense to piggyback the development of future ubiquitous computing application to WWW technology.

2.1 Communication Primitives for the WoT

WoT 2011, June 2011; San Francisco, CA, USA

Almost a decade ago Saif, et. al. [1] summarized the challenges for communication in ubiquitous computing. At the time he argued against RPC style interaction in middleware systems like CORBA. During the following decade a huge amount of specialized ubiquitous middleware systems (such as [2], [3]) were develop that take into account those consideration. The WoT takes a new pragmatic approach at enabling ubiquitous application using the world wide web as an accellerator leaving aside the ballast of other middlewares but also often being limited by the way the current WWW works. In the following we like to summarize how existing WoT platforms can provide a communication platform for ubiquitous computing application.

2.1.1 Interface Definition

[1] stresses the need of loose coupling between components and component development in ubiquitous computing. This analyis actually directly reflects the motivation for RESTful interfaces in the world wide web. The "HATEOAS" principle distinguishes REST style architectures from traditional client server system. This loose coupling between interfaces using self-descriptive interfaces can also be found in early ubiquitous computing paradigms like ConCom [4]. Fieldings "software design on the scale of decades" is a perfect match to the problem of ubiquitous computing technology. As stated by [5] it makes REST a good choice for building ubiquitous application in an internet of things. Sensor.Network[6] proposes a sensor data sharing platform that can upload and retrieve sensor data through RESTful interface. Pachube[7] is also a data aggregation and distribution platform with RESTful access. The *uBox* can be seen as a "pocket" version of such systems.

2.1.2 Device Control, multimedia communication and Multi-party interaction

However, more flexible interaction patterns that are needed to support ubiquitous computing application. While the classical Web is hypertext and document oriented in ubiquitous computing common requirement to receive streams of content [1]. Such media streams are not only sound or video but any kind of data that can be digitalized using sensors. This data needs to be streamed live towards multiple sinks to usefully aggregates, e.g. for context and activity recognition.

In the document based WWW Atom and RSS feeds are used for aggregating streams of new data. Such batched client pull notification via HTTP can also be found in [6] or [7]. The interactive Web has successfully used AJAX technology to send data asynchronously to the client. However, on a technical protocol level AJAX requires heavy interaction between client and server imposes a problem on ressource constrained devices when serving multiple clients. The *uBox* provides a transparent layer that technologically decouples the systems while allowing AJAX style interaction.

An alternative a data base often serves as a buffer for to give access to data from IoT devices. In the end databases founded approaches, which eagerly "cache" sensing data, like [6], SenseWeb[8] or SensorBase[9] will also have scaling issues when it comes to dynamically distributing data. *uBoXing* scales by adding more *uBoxes*.

2.1.3 Logical and physical mobility

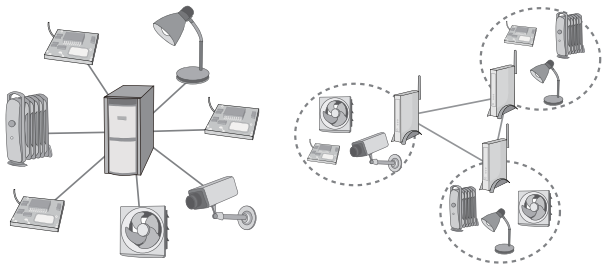

Figure 2: Architectures of WWWoT: a)All the information of resources from world wide are stored in centralized server. b) uBoxing:Local nodes contain their own resources and make a network among them.

[1] points out is that one of the key aspects of ubiquitous systems is mobility. The World Wide Web is built around static availabilty of ressources. Wireless system and especially things however move around physical spaces without any network barriers with varying link properties. Systems like dynamic DNS systems and worldwide routing can to some extend hide the physical mobility of ressources, but the effects of mobility are visible in the real world and should thus be in the WoT.

Ubiquitous computing location and communication based on location as context is an important factor. Many applications are e.g. based on proximity or other types locality. Resources within the WoT need namespaces based on a contextual view that a static hierarchy like the current WWW cannot provide. E.g. the temperature for the room as a ressource can be provided by all the things in the room an application should be able to address the room as an abstract ressource without querying all devices explicitly.

uBoXing enables distributed, active namespaces. Following the plan 9 paradigm [10]. "every resource in the system, either local or remote, is represented by a hierarchical file system; and a user or process assembles a private view of the system by constructing a file name space that connects these resources".

2.1.4 WoT beyond 2011

In order to show how our work fits not only fits the needs of ubiquitous computing but can be also viewed in the scope of developments going on in the World Wide Web in general we picked 3 item from the current OReilly Radar's Watchlist [11]:

Real time data As the WWW is becoming more dynamic with content created automatically and being continously update there needs to be a paradigm shift for indexing and searching. With Percolator [12] Google moves their efforts to process data in the web from document batch processing towards incremental data updates. The *uBox* provides a simple yet scalable way to introduce real time data processing on a local level eliminating the necessety for centralized sytems and extend it via the *uBoXing* architecture.

The return of P2P "Many factors are coming together to drive a search for a new architectural model: the in-

ability of our current provider paradigm to supply the kind of network we'll need in the next decade" [11]. The WoT is such a network. In P2P systems everybody can contribute to the new WWW by adding local resources without subscribing to the data gathering schemes of internet scale service providers. *uBoXing* is new way to share sensor and actuatuator ressources in a P2P fashion.

The meaning of privacy Locality of data also becomes important as the WWW becomes aware of privacy. If someone hacks into a centralized server and can analyze and intrude the physical world of millions of people a crucial barrier is broken. For this reason, people will hesitate to register sensors or actuators to a WoT Platform and the real WoT will not happen. The *uBoXing* concept clearly addresses those concerns in the architecture by building on a local system where ressources are freely accessible in the local domain but where information, e.g. from aggregated data can be exposed to the *uBoXing* network.

3. DISTRIBUTION OF RESOURCE MANAGEMENT

From our analysis we have seen that neither exposing devices directly to the WWW nor using centralized systems (Figure2a.) are appropriate structures for a World Wide WoT. For these reasons, we propose a distributed resource management architecture to create World Wide WoT whose topology is Mesh like as shown in Figure2b. In this architecture, there are stand-alone local WoT platform for each WoT domain. This gives an aspect of local resource management to each WoT. Then we create a network of those local WoT platforms with an add-on middleware. This method decentralizes the resource management and solves the problem of total dependency.

To enable this concept, we propose the architecture that creates World Wide WoT with two steps. The first step is to manage local WoT environments with a stand-alone platform, which is called *uBox*. The second step, we should install an add-on middleware to interconnect local WoT platforms and create World Wide WoT, which we call *uBoXing*. In this section, we will describe how this model contributes to WoT environment.

3.1 Aspect of Local WoT Management

The *uBox* is nothing to do with World Wide WoT by itself. This platform is only for managing one domain of WoT for internal use. Initially WoT are built to make a certain environment smart, such as an office or a house, based on special needs. *uBox* supports this feature of WoT. When administrators register their networked sensors or actuators to this platform, the users or developers within the network can discover them and utilize them with a simple unified interface. They can register their internal administrative information to the platform since *uBox* is totally under their control and external user will not access the platform server directly. This helps simple private application development.

Now, in WoT environment, many resources, which are either sensor or actuator, are connected to the network and can be manipulated through the network. However, each resource has own way of access. Application developers have to read a document for each resource to know how to access it. Even before reading documents, he has trouble finding what are installed in the environment. If application developers have a private WoT platform that provides them a simple unified interface to search, and access the resources in their environment, their amount of task to create a WoT application will be much reduced. In this way, WoT becomes a local infrastructure, and the application developers can focus on application itself.

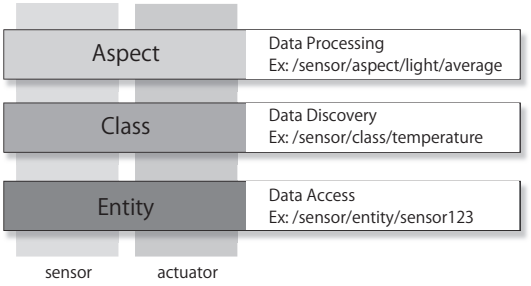

Figure 3: Layer Abstraction of WoT. Entity layer provides direct node access. Class layer provides node discovery. Aspect provides node discovery and data processing of discovered nodes.

3.2 Network among Local WoTs

uBox can be interconnected each other(*uBoXing*) by add-on middleware, which we call global gateway, although they are stand-alone by themselves. Global gateway belongs to the same network as local WoT platform but port is opened for the Internet. This add-on has two functions, which are 1) to control external access to the sensors and actuators that is managed by attached WoT platform, and 2) to route the users' request for appropriate local WoT platform placed somewhere in the world.

As for the access control feature of global gateway, it accepts requests from external network and bridge the request and response. In request bridging, accessibility to the node is according to the information that comes from its local WoT platform, which are open, close, authenticate. As for the routing feature, it creates overlay network of global gateways over the Internet with utilizing physical distance as a metric. Then when it receives a request from user, the global gateway will deliver the request to gateways that contains their target data via the overlay network.

Using *uBox* and attaching global gateway as an add-on middleware for *uBoXing*, we can create WWW WoT with local WoT centric approach.

4. SYSTEM DESIGN

This platform is based on an layered abstraction as shown in Figure3 and each layer can be manipulated via RESTful interface. Users or applications can access each layer independently. They can also specify the target scope from whether local or global. The URL format is http://[server]/[local/global]/[sensor/actuator]/[layer]/[parameters]. Each layer will be described as follows.

4.1 RESTful Access To Resources

4.1.1 Entity: Direct Access To A Resource

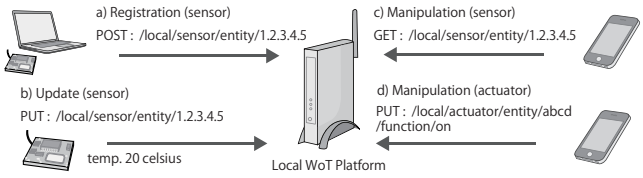

Figure 4: Requests example for entity layer. Users can register, update, manipulate and delete with RESTful url.

```
POST:http://wot.teco.edu:8082/local/actuator/entity/light12345
{    "latitude":"49.00",
     "longitude":"8.38",
     "permission":"0",
     "description":"desk light of the room 203.",
     "tags":["room","desk"],
     "functions":{
          "on":{
               "url":"http://actuator.teco.edu:5001/power/000D6F00007294BB",
               "method_type":"GET", "actuator_type":"light","permission":"0",
                "description":"turn on the power",
               "parameters":{"time":{"default":"0", "description":"time delay."}}
          },"off":{...}
     }
}
```

Figure 5: JSON for registering an actuator.

The bottom layer of the WoT access abstraction is called "Entity", which enables direct access to the resources with specifying their resource id. Entity layer supports registration, update, manipulation, and deletion of information of a resource.

- **Registration:** To register the resources, a user should create HTTP request via POST with parameters that are formed as JSON to entity layer. Figure5 shows the JSON request to register actuator that is for private use, located to coordinates (49.00,8.38) and tagged by "desk" and "room". Tag is used for searching. The key that sensor registration request does not have is "function", which defines how to actuate the resource. In this case, the request defines functions named "on" and "off". In the function named "on", they specify resource uri, method type, necessary parameters, and accessibility. The actuators are supposed to have also RESTful support, either they have native RESTful support or they have proxy which enables RESTful access to the actuator. Resource registration is only allowed locally.

- **Update:** To update resource information, you should use PUT method with parameters that are also formed as JSON. The parameters are the same format as the registration, only with the information that you want to update. As for sensors, sensor data are consider to be attributes of sensor. Therefore, you make a PUT request with sensor data into the url that is shown in Figure4b to insert new sensor data. JSON format for data insertion is shown in Figure6. Resource update is also allowed only within internal network.

```
PUT:http://wot.teco.edu:8082/local/sensor/entity/1.2.3.4.5.124
{    "data":{
          "temperature":{
               "unit":"celsius",
               "value":"20"
          }
     }
}
```

Figure 6: JSON for updating sensor data.

- **Manipulation:** Resource manipulation means "acquiring data" for sensors and "evoking a function" for actuators. For sensors, users need to create GET request. The query format is shown in Figure4c. If optional parameter "cache" is set false, the server waits replying the response until next data comes. Therefore, event driven application can be developed over this platform. For actuators, the user need to create PUT access with required function specific parameters. PUT request such as shown in Figure4d will invoke the function "on", which is registered by the request shown in Figure5. According to RESTful concept, we have decided to use PUT method for invoking a function since it changes a status of the resource or situation of the environment. This feature can be applied to both localy and globally.

- **Deletion:** To delete a resource, the user should make a DELETE request to the url, which is the same url as registration. This will delete device information, and corresponding data as well. This request is only used within the internal network.

4.1.2 Class: Discovering Resources

We named the second layer of abstraction "Class". This layer supports resource discovery. Users can discover the sensors with specific type of data or actuators. For example local/class/actuator/light will discover actuators that control the light in the local area. To collect temperature sensors tagged as "outside" and located within 1000m radius circle from coordinates(49.00, 8.38), the user should make GET request to the url shown in Figure7a. This layer only supports GET requests. Requesting for global resource in this layer and Aspect layer, which is described next, will follow the sequence diagram shown in Figure.8. First, 1)an application sends the HTTP/GET request to a uBox in their home. 2)the uBox requests for the IP addresses of uBox in a target area to device discovery network. And 3) they the uBox sends requests for the individual remote uBoxes to collects target datas.

4.1.3 Aspect: Processing Resources

The top layer of the abstraction is named "Aspect", since this layer processes multiple data and generates data that represents their aspect. For example users can create GET request such as shown in Figure7b. This request will collect temperature sensors tagged as "outside" and located within 1000m circle from coordinates(49.00, 8.38) and process average value for each unit. The procedure of your own can be added to your *uBox* with posting JavaScript.

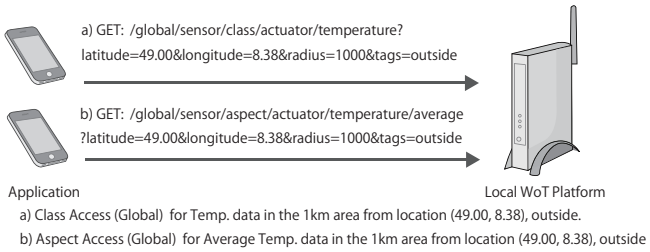

Figure 7: Requests example for class, and aspect layer. Users can find and process global sensor data with conditions.

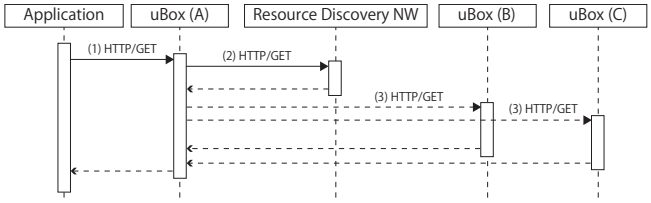

Figure 8: Sequence Diagram of Request Process Procedure for the World Wide Request

4.2 Global Gateway

We call the add-on software Global Gateway. This gateway exchanges geographical location of *uBox*, and creates overlay network on the Internet utilizing their physical distance as a metrics. Therefore, users can ask for the global sensor data to his local gateway as well with global gateway. If user requests for temperature data around his location, it forwards the requests to its global gateway. Then global sensor network gateway passes the requests to global gateway that is within the target area or the closest WoT platform. The local platforms pass the request forward for N times and search the target sensors. With this way, users can request sensor data in any part of the world with the same manner with local requests.

5. SMART TECO ENVIRONMENT

SmartTecO is a WoT environment that is running in our lab. The sensors and actuators that we utilize are shown in Figure9. Figure9a is a device called "uPart"[13], which is a tiny wireless sensor node with temperature, light, and movement sensors implemented. Figure9b is "d-bridge"[14], which is a programmable base-station for uPart. D-bridge receives uPart packet and forward them to *uBox* via HTTP. Those devices are made in TecO. Figure9c is a device called "Plugwise". This device senses power consumption, as well as controls on/off state of the power plug via ZigBee. A device in Figure9d is "FHT80b". This is an actuator that controls heater via wireless network. We placed a PC that bridges commands to Plugwise and FHT80b from HTTP, since they are commercially-supplied and both do not have RESTful interface.

5.1 Management Web-Interface

We have created browser-based external user interface for *uBox*. This interface creates variety of requests that are ex-

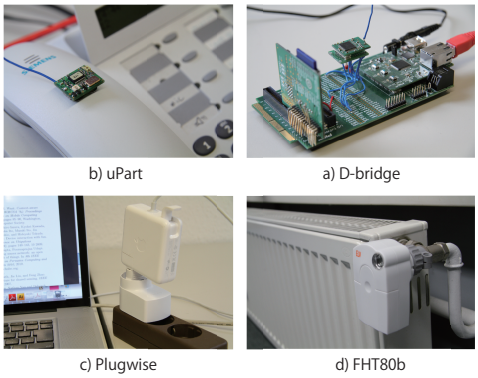

Figure 9: Resources at SmartTecO. a)D-bridge, b)uPart, c)Plugwise and d)FHT80b

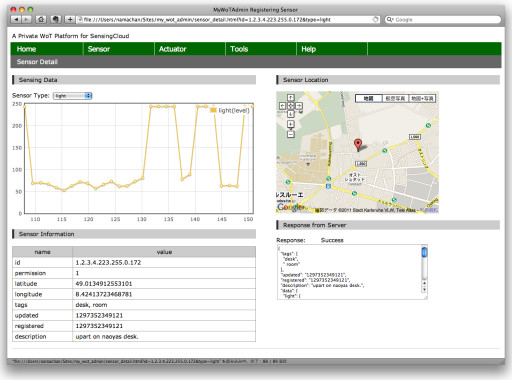

Figure 10: Screen dump of uBox interface.

plained above with web interface with Ajax. Figure10 shows the screen shot when a user checks the property of a sensor node. The sensor static information with dynamic real-time updating data plot is shown on the web browser. Also, the user can control actuator as well through this interface. This interface is independent from the core software. It only calls the URLs that the platform offeres to normal users. Therefore, this interface is also one of applications built on the platform. Mainly this web interface is using Ajax technologies, such as a graphing library, a mapping component and Ajax library. As this web interface, which is mash-up existing techniques, shows that our approach fits together well to the other state-of-the-art technologies. This also confirms that we enabled loose coupling between components with the platform.

5.2 Smart Laboratory

The laboratory at TecO has traditionally been a living lab for ubiquitous technology. Some smart applications have become essential part of the infrastruction. One of those applications was a smart alarming system for our hardware lab. It would alarm the user and proactively turn of our left-on soldering equipment when nobody was in the room. The old wired hack was running for years until we moved our lab equipment. While rewiring seemed too difficult, the

application seemed to a perfect first candidate to test our World Wide Wot. We started with adapting the ClickScript IDE (clickscript.ch) to support our dynamic discovery mechanisms and to use dynamically generated modules. By including both uParts to detect movement around the table and the plugwise devices to measure and control our soldering irons, we can easily create the old application with virtual instead of physical wiring.

Since the first prototype idea we have started (re-)building many existing and new application of on top of WoT technology:

- The smart meeting room can inform anybody who is standing in front of the closed door if there is really a meeting happening. It also can automatically post a 30minute reservation to our meeting room booking system.
- The smart mailbox will email the user if (postal) mail has been dropped in his mailbox on campus (otherwise a 10min walk to check).
- The smart heating system can set all thermostats in the room via the FHT80b based on the average of personal preference of all people in the room (Professor overrides).

5.3 Mobile Sensor Applications

Mobile applications in World Wide WoT can interact with arbitrary sensors around the device. For example, a user can have a fine-grained weather forecasting service around him. The application keep requesting for the data of rain sensor within 1km area around the user's location. Then, when the application finds the raining spot within the area, he will be notified and the map that shows current raining area appears. By watching the map that is changing dynamically, he will predict when it will start raining where he is now.

6. CONCLUSION AND FUTURE WORK

We proposed a distributed resource management architecture for interconnecting WoT with "local platform" and "global gateway". Local WoT platform enables ubiquitous computing environment with a classical UbiComp approach, which is interacting with its surroundings, rather than a world wide services. However, global gateway as an add-on middleware enables to interconnect the local WoT platforms and creates a World Wide WoT, which enables for applications to interact with multiple WoT domains. This gives applications on WoT mobility and proximity. This approach creates World Wide WoT with leaving resource management to local domain.

We build a first proof of concept prototype that allowed us to build ubiquitous computing application in our lab on top of Web-of-Things technology. Our future work is considering better methods to create the overlay networks of local WoT platform. There are other promising works in that like pubsubhubbub or telehash that integrate webtechnology and P2P networks. Platforms like netkernel provide a powerful tool for in-network processing. In the future we like to establish an open distributed communication framework based on World Wide Web technology for building both local and global WoT platforms.

Acknowledgement

This work was partially funded by the German Ministry of Research and Education as part of the Aletheia project.

7. REFERENCES

[1] U. Saif and D.J. Greaves. Communication primitives for ubiquitous systems or RPC considered harmful. In *Distributed Computing Systems Workshop, 2001 International Conference on*, pages 240–245, 2001.

[2] C.K. Hess and R.H. Campbell. A context-aware data management system for ubiquitous computing applications. 2003.

[3] Jini. http://www.jini.org/.

[4] A. Krohn, M. Beigl, C. Decker, P. Robinson, and T. Zimmer. *ConCom: A Language and Protocol for Communication of Context*. Universität Karlsruhe, Fakultät für Informatik, 2004.

[5] D. Guinard, V. Trifa, and E. Wilde. A resource oriented architecture for the web of things. *Proc. of IoT*, 2010.

[6] Arshan Poursohi Vipul Gupta, Poornaprajna Udupi. Early lessons from building sensor.network: an open data exchange for the web of things. In *8th IEEE International Conference on Pervasive Computing and Communications, PerCom 2010*, 2010.

[7] Patcube. http://www.pachube.org.

[8] Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao. Senseweb: An infrastructure for shared sensing. *IEEE MultiMedia*, 14(4):8–13, 2007.

[9] Mark Hansen Gong Chen, Nathan Yau and Deborah Estrin. Sharing sensor network data. Technical report, Center for Embedded Network Sensing, March 2007.

[10] R. Pike, D. Presotto, K. Thompson, H. Trickey, and P. Winterbottom. The use of name spaces in plan 9. In *Proceedings of the 5th workshop on ACM SIGOPS European workshop: Models and paradigms for distributed systems structuring*, pages 1–5, 1992.

[11] 2011 watchlist: 6 themes to track - O'Reilly radar. http://radar.oreilly.com/2011/01/2011-watchlist.html.

[12] Frank Dabek Daniel Peng. Large-scale incremental processing using distributed transactions and notifications. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[13] M. Beigl, A. Krohn, T. Riedel, T. Zimmer, C. Decker, and M. Isomura. The upart experience: building a wireless sensor network. In *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, pages 366 –373, 2006.

[14] Dawud Gordon, Michael Beigl, Martin Alex, and Er Neumann. dinam: A wireless sensor network concept and platform for rapid development. In *the Seventh International Conference on Networked Sensing Systems, INSS 2010*, 2010.